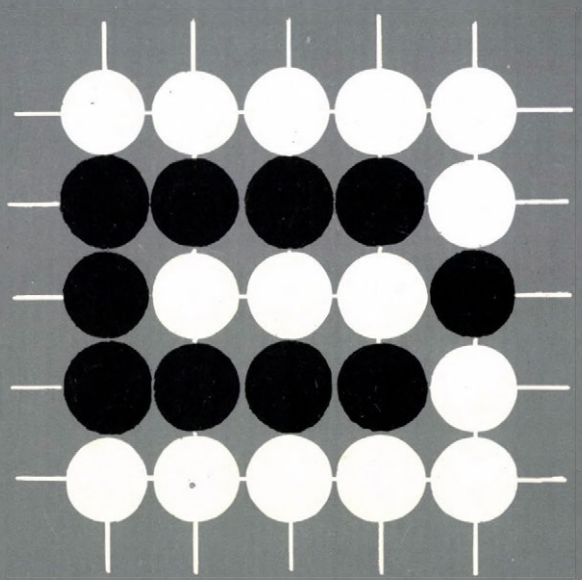
közlemények

35/1986

MTA Számítástechnikai és Automatizálási Kutató Intézet

Budapest



Magyar Tudományos Akadémia
Számítástechnikai és Automatizálási Kutató Intézete
Computer and Automation Institute, Hungarian Academy of Sciences

K Ö Z L E M É N Y E K

T R A N S A C T I O N S

35/1986

Felelős kiadó:

DR. KEVICZKY LÁSZLÓ

ISBN 963 311 224 9
ISSN 0133-7459

C O N T E N T S

T A R T A L O M J E G Y Z É K

old.

MTA SZTAKI Közlemények 35/1986 p. 7-30

ЗАДАЧА СНАБЖЕНИЯ РЕГИОНА НЕФТЕПРОДУКТАМИ ПРИ НАЛИЧИИ ПРОДУКТОПРОВОДА ДЛЯ ПОСЛЕДОВАТЕЛЬНОЙ ТРАНСПОРТИРОВКИ НЕСКОЛЬКИХ НЕФТЕПРОДУКТОВ

Инзелт Петер
Исследовательский институт вычислительной техники и автоматизации ВАН, г. Будапешт

I. ВВЕДЕНИЕ

Транспортировка больших количеств жидких нефтепродуктов наиболее экономично может решаться с помощью продуктопровода. Эксплуатационные расходы продуктопровода в 50-100 раз ниже транспортных расходов автотанкеров или железнодорожного транспорта. В развитых странах были сооружены разветвленные сети продуктопроводов и начиная с конца 70-х годов в нашей стране также быстрыми темпами создаются продуктопроводы.

Возможность дешевой транспортировки различных нефтепродуктов по продуктопроводу значительно может изменять регионы снабжения отдельных нефтеперерабатывающих комбинатов. Если НПЗ и большие склады нефтепродуктов соединены между собой продуктопроводом, обладающим достаточно большей пропускной способностью, буферные запасы по продуктам или компонентам, необходимые для покрывания сезонального колебания спроса, не обязательно должны храниться там, где они вырабатывались. Если не фиксируются границы региона снабжения отдельных НПЗ - как это имеет место в настоящее время - можно получить отличающиеся результаты при планировании количества перерабатываемой нефти по НПЗ, по ассортименту продуктов НПЗ и стратегии хранения буферных запасов сезонального колебания спроса.

Необходимо уже здесь отметить, что включение продуктопроводов в модели планирования снабжения регионов (страны) является исключительно сложной, практически нереализуемой задачей. Дело в том, что пропускную способность продуктопровода нельзя характеризовать каким то числом: как покажем, она очень

сильно зависит от того, какие продукты куда, в какой последовательности и в каких количествах транспортируются, кроме того, существует ряд дополнительных условий реализации определенного графика транспортировки. Данная проблема не может быть решена в рамках модели линейного программирования, которая обычно применяется для планирования производства нефтепродуктов и снабжения региона: управление работой продуктопровода необходимо анализировать параллельно с задачей планирования производства нефтепродуктов.

Наличие сети продуктопроводов влияет также на проектирование новых капиталовложений, на территориальное распределение крупных складов для снабжения областей и на соответствующие емкости хранения по продуктам. Как видно будет, любое капиталовложение, изменение в окрестности продуктопровода влияет на условия работы всего комплекса, то есть альтернативнве решения должны анализироваться не по объектам, а с точки зрения всей системы, включая условия работы продуктопровода.

Номинальную пропускную способность продуктопровода можно использовать только при транспортировке одного продукта по нему, транспортировка нескольких продуктов всегда снижает эффективную пропускную способность продуктопровода. Ввиду дешевой эксплуатации продуктопровода, следует стремиться к тому, что максимально использовать его пропускную способность, в то же время реализация графика поставки в окрестности максимальной нагрузки продуктопровода является сложной и ответственной задачей. Упрощения в процессе составления графика, случайные изменения в ходе реализации могут привести к сложным ситуациям, поэтому оперативные решения диспетчера продуктопровода должны поддерживаться математическими средствами.

2. ОСНОВНЫЕ ПОНЯТИЯ И ФОРМУЛИРОВКА ЗАДАЧИ ПЛАНИРОВАНИЯ РЕГИОНАЛЬНОЙ СИСТЕМЫ СНАБЖЕНИЯ

2.I. Региональная система снабжения

Региональной системой снабжения (РСС) называется совокупность производственных мощностей, складов и транспортных возможностей, обеспечивающая снабжение данного региона жидкими (светлыми) нефтепродуктами, как автобензины различных марок, бензины для химической переработки, газойлы.

РСС состоит из узлов и пунктов потребления, которые соединены между собой (в одно или обоих направлениях) путями перевозки.

Узел характеризуется тем, что он является источником, далее возможным поглотителем большого ассортимента нефтепродуктов (в том числе нетоварных продуктов, компонентов, и обладает значительными буферными мощностями. С точки зрения путей перевозок, принятых во внимание в РСС, узел обладается возможностью отгрузки. Пункт потребления играет роль только поглотителя с ограниченной буферной мощностью, он не обладает возможностью отгрузки товаров к узлам или другим пунктам потребления. В пункте потребления встречаются только товарные нефтепродукты (их определенный ассортимент).

Пути перевозки обеспечивают транспортировку нефтепродуктов от узлов к пунктам потребления и между узлами. Путями перевозки являются продуктопровод, железная дорога, автотранспорт, водный транспорт.

Транспортировка по продуктопроводу значительно (в 50–100 раз) дешевле другим видам транспорта, следовательно, следует максимально использовать его пропускную способность. В то же время следует отметить, что пропускная

способность продуктопровода зависит от конкретного графика постовок, то есть от того, какие продукты, в каком количестве и куда поставляются. С небольшим упрощением реальной ситуации можно сказать, что такой зависимости в случае других видов транспорта не наблюдается, если их пропускная способность задается в т. км и заправочные устройства в НПЗ имеют достаточную производительность. Следовательно, что с точки зрения транспортируемых партий, пути перевозки можно разделить на зависимые (продуктопровод) и независимые (железнодорожный-, водный- и автотранспорт).

2.2. Задача планирования производства РСС

При планировании производства РСС следует дать ответ на следующие основные, связанные друг с другом вопросы:

I. Из какого узла (каких узлов) должны поставляться продукты в конкретные пункты потребления, пути перевозки, момент времени и количества этих поставок;

2. Для оптимального снабжения региона какое количество нефти должно перерабатываться в заданном интервале времени и в каких режимах в узлах системы (НПЗ), какие дополнительные перевозки (компонентов, продуктов) необходимы.

Под критерием оптимальности принимается например минимизация суммарных расходов на переработку, транспортировку и хранение нефти и нефтепродуктов.

Рассмотрим только детерминированную задачу, не забывая, что большинство исходных данных носит вероятностный характер и имеет смысл рассматривать также вопросы надежности выполнения плана. Предполагается, что заданы потребности по каждому виду продуктов в каждом пункте потребления на период планирования в виде кусочно-постоян-

ной функции, известны запасы в начальном моменте планового периода, все технологические параметры и ограничения (например, емкости резервуаров по продуктам, производственные мощности, модели переработки нефти (выход компонентов) и модели смешения, и т.д.). Известны также различные коэффициенты стоимости (например, удельная стоимость транспортировки нефтепродуктов по различным путям перевозки, стоимости переработки нефти на различных НПЗ, - если они отличаются -, и т.д.).

С точки зрения снижения суммарных расходов по всей системе определяющим является снижение расходов на транспортировку нефтепродуктов, то есть именно оптимальное распределение нефтепродуктов. Стоимость переработки или хранения нефти на различных НПЗ обычно мало отличается друг от друга, расходами на транспортировку самой нефти тоже можно пренебречь, потому что она транспортируется по трубопроводу.

Данная задача должна решаться при условии 100 %-ного покрывания спроса. Ввиду того, что пропускную способность продуктопроводов нельзя характеризовать однозначным числовым значением и описание работы продуктопровода - как будет показано - исключительно сложное, в модель линейного программирования, обычно использованная для решения планирования производства РСС, нельзя включить продуктопроводы. Необходимо прибегать к итеративному решению: или определить пределы пропускной способности продуктопроводов при предварительно заданных спросах и стратегиях их покрывания и проверить экономичность различных вариантов (причем количество возможных стратегий очень большое) или рассчитать план с использованием предварительной оценки пропускной способности продуктопровода и проверить реализуемость полученного плана. В обоих случаях можно получить только субоптимальное решение.

По моему мнению, второй вариант является более приемлимым. Принимая во внимание размерность реально решаемых задач, в этапе календарного планирования необходимо разбивать проблему по подсистемам. Несомненно, требуется отдельная модель для узлов (НПЗ) и тогда имеет смысл модель календарного планирования работы продуктопровода, не включающего в себя узлы (для этой модели узел является неограниченным источником - поглотителем, или узел характеризуется набором ограничений). В результате решения глобальной задачи планирования РСС уже имеется основная стратегия переработки нефти и снабжения, что определяет объем поставок между узлами и прикрепляет пункты снабжения по конкретным продуктам к конкретным узлам (и определяет план отдельных НПЗ, что в данном моменте вне сферы наших рассуждений). Теперь необходимо приступить к проверке реализуемости этого плана по продуктопроводам.

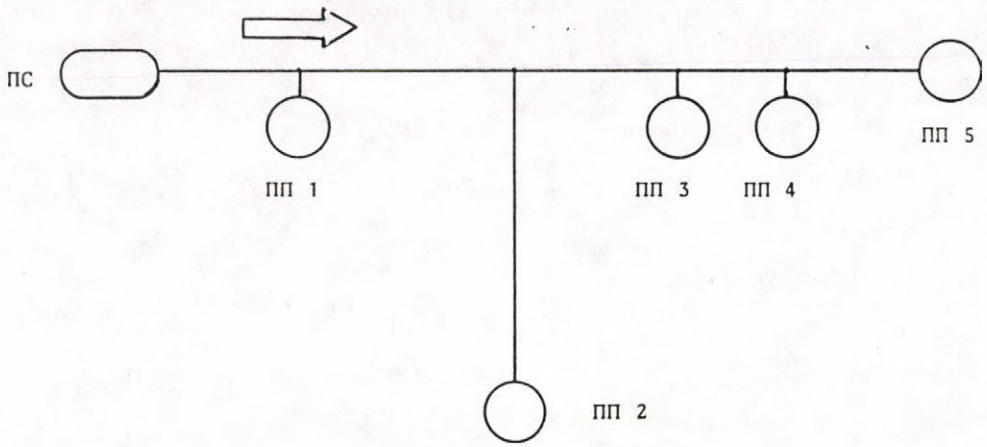
3. ЗАДАЧА УПРАВЛЕНИЯ ПРОДУКТОПРОВОДОМ ДЛЯ ПОСЛЕДОВАТЕЛЬНОЙ ТРАНСПОРТИРОВКИ НЕСКОЛЬКИХ НЕФТЕПРОДУКТОВ

Можно различать три основных типа продуктопроводов для последовательной транспортировки нескольких продуктов к нескольким пунктам потребления (рис. I):

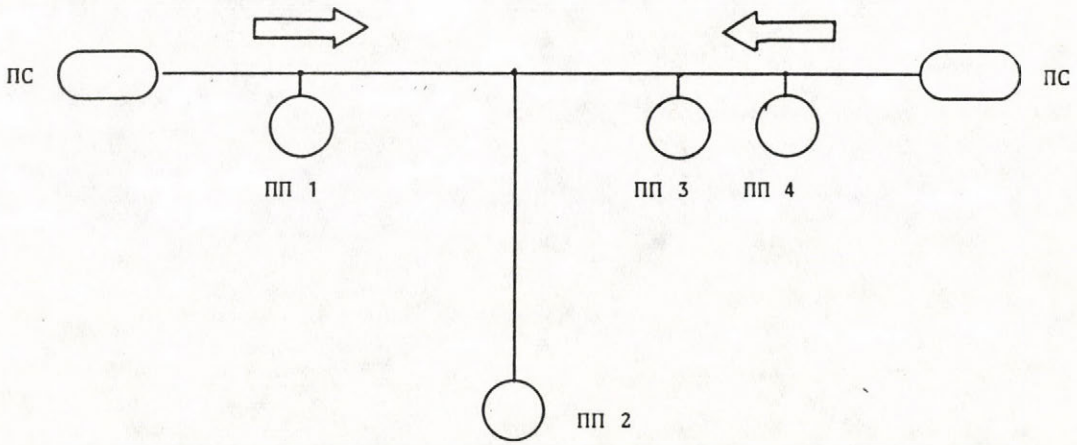
- транспортировка в одно направление (поставка к нескольким пунктам потребления из одного узла)
- транспортировка в два направления (в обоих концах продуктопровода находятся узлы, к пунктам потребления можно поставлять продукты из любого узла, возможна транспортировка нефтепродуктов - или компонентов - между узлами)
- параллельные продуктопроводы с возможностью снабжения пунктов потребления от любого продуктопровода (возможна транспортировка в одно или обоих направлениях, продуктопроводы могут иметь разные диаметры, условные давления).

ПС - Пункт снабжения ПП - Пункт потребления

1/а. Транспортировка в одно направление



1/б. Транспортировка в два направления



1/в. Транспортировка в два направления, два параллельных продуктопровода

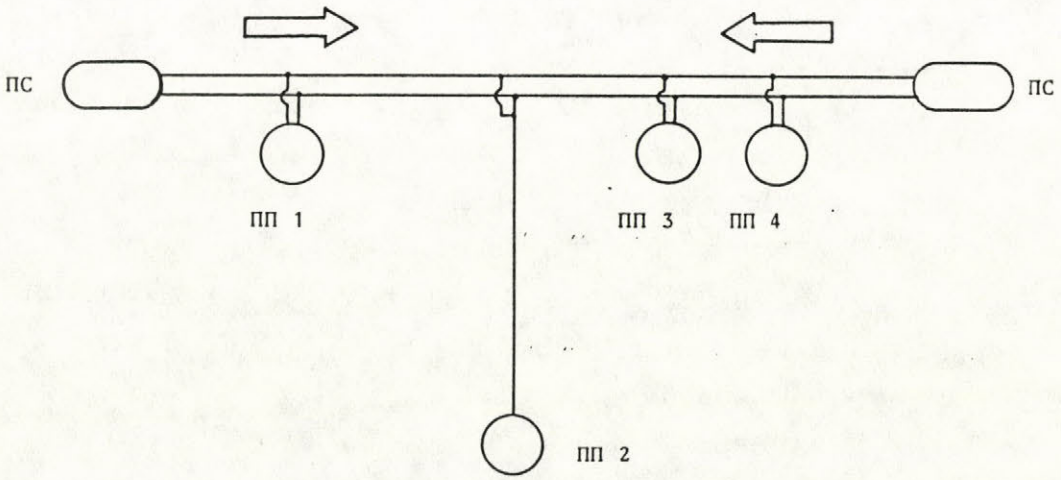


Рис. 1. Основные типы систем регионального снабжения с продуктопроводом

Любая сеть продуктопроводов разбивается узлами на участки вышеприведенных типов, то есть достаточно анализировать особенности работы этих участков.

Рассмотрим самый простой случай, транспортировку нефтепродуктов в одно направление.

3.I. Краткое описание технологии транспортировки

Продуктопровод для транспортировки нефтепродуктов в одно направление (рис. I.а) исходит из узла, а вдоль его размещены пункты потребления. Часть пунктов потребления находится вблизи продуктопровода (можно пренебречь объемом участка, соединяющего продуктопровод с пунктом потребления), они называются ответвлениями, другие находятся на значительном расстоянии (I5-50 км) от продуктопровода, их будем называть разветвлениями. Последние теоретически могут рассматриваться, как самостоятельные продуктопроводы с одним пунктом потребления (при этом пункт разветвления должен рассматриваться узлом, что очень усложняет задачу).

Работа продуктопровода характеризуется следующими основными положениями:

- продуктопровод всегда наполнен нефтепродуктом;
- давление должно сохраняться в заданном диапазоне, что ограничивает количество одновременно наполняемых пунктов потребления (отбора продукта из продуктопровода);
- транспортировку можно прекратить только при условии, что трубопровод наполнен однородным продуктом, потому что иначе повторный пуск транспортировки приводит к интенсивному перемешиванию на границе фаз и в связи с этим к значительным потерям;
- линейная скорость перемещения продуктов находится в диапазоне 4 км/час;

В дальнейшем системой снабжения будем называть совокупность продуктопровода, узла, пунктов потребления и независимых путей перевозки, соединяющих узел с этими же пунктами потребления.

Пусть известны следующие исходные параметры системы снабжения:

- количество пунктов потребления (n),
- количество продуктов, транспортируемых по продуктопроводу (m),
- параметры продуктопровода (диаметр, длина участков между ответвлениями и разветвлениями, и т.д.),
- емкости резервуаров в каждом пункте потребления по продуктам ($м_i^j$, i=1,2,...,n; j=1,2,...,m); $м^3$,
- запасы в каждом пункте потребления в момент времени t_0 по продуктам (q_{io}^j), $м^3$,
- спрос в каждом пункте потребления в интервале времени T по продуктам (z_i^j (T); $T=\{t_0,t_v\}$), $м^3$/час;
- возможность доставки продуктов по независимым путям перевозки (диапазон интенсивности возможной поставки λ_i^j), $м^3$/час.

Вопросами прогностизирования спроса по пунктам потребления и продуктам здесь не занимаемся. В практике потребители (заправочные станции, МТС, колхозы, и т.п.) обычно прикреплены к заданному пункту потребления. Отметим только, что решение задачи управления продуктопроводом носит субоптимальный характер и потому, что совершенно не исключена возможность реализации более оптимального плана при относительно небольших изменениях спросов по пунктам потребления (при изменении прикрепления потребителей даже при условии, что расстояния на локальные перевозки и соответствующие расходы немного увеличиваются). Вообще говоря,

если по конкретным пунктам потребления регулярно наблюдаются "узкие места", целесообразно проверить систему прикрепления потребителей. В то же время, непосредственно стыковать эти задачи, по моему мнению, нельзя.

Можно предполагать, что стоимость транспортировки не зависит от вида продукта и каждый пункт потребления прикреплен только к одному виду независимого транспорта. Если речь идет об автотранспорте, продукты не должны транспортироваться в пункт потребления, а могут поставляться непосредственно к его потребителям, но эти случаи не различаются.

Пусть примем следующие упрощения в связи с эксплуатацией системы:

- спрос предполагается постоянным во времени в интервале T по пунктам потребления и продуктам, то есть

$$z_i^j(T) = z_i^j = \text{const}$$

- максимальная скорость отбора продуктов в пунктах потребления зависит только от диаметра соединительного трубопровода, то есть является заданной константой (w_i, $м^3$/час). Объемная скорость продукта в самом продуктопроводе зависит от того, что в данный момент времени в какие пункты потребления отбираются продукты, причем объемная скорость продукта в продуктопроводе ограничена для избежания интенсивного перемещения на границах фаз,
- ограничено общее количество одновременно обслуживаемых пунктов потребления (не более 2-3),
- узел рассматривается неограниченным источником и предполагается, что по независимым путям поставки можно поставлять любое количество нефтепродуктов в пункты потребления, ограниченное только пределом интенсивности приема на пунктах потребления (общее ограничение по количеству автотанкеров, вагонов не принимается во внимание. При

необходимости ограничение на общую пропускную способность независимых путей поставки можно включить в модель).

3.2. Некоторые характеристики системы снабжения

Рассмотрим возможные случаи поставки j-того продукта в количестве g_i^j в i-той пункт потребления.

Знаем, что продуктопровод всегда наполнен нефтепродуктом. Если в данный момент транспортировка по продуктопроводу не идет, возможны два случая. Если при i-том пункте потребления находится j-тый продукт, поставку можно начать немедленно, только необходимо включать насосы. Конечно, предварительно надо принимать решение о том, какой продукт нагнетать в трубопровод с узла!

Если при i-том пункте потребления находится отличающийся от j-того продукт, то ситуация другая. С узла можно начать нагнетание j-того продукта, но необходимо обеспечить опорожнение участка между узлом и i-того пункта потребления. Для этого необходимо иметь свободные резервуары. Продукт, находящийся в продуктопроводе можно распределить произвольно между пунктами потребления всей системы снабжения (так как транспортировка не шла, содержимое продуктопровода должно быть однородным), в количестве, не менее объема участка между узлом и i-тым пунктом потребления. Следует иметь в виду, что интервал времени доставки j-того продукта от узла к i-тому пункту потребления зависит и от распределения содержимого продуктопровода, так как скорости отбора по пунктам потребления могут отличаться!

Если в момент возникновения данного спроса продуктопровод работает, ситуация аналогична с тем, что возможно дополнительное запаздывание ввиду того, что не всегда возможно или целесообразно прерывание текущей операции.

Из изложенного видно, что продуктопровод является своеобразным, ограниченным ресурсом, который занимается удовлетворением каждого спроса на определенный интервал времени. Время между проявлением и удовлетворением конкретного спроса зависит и от состояния продуктопровода и всей системы снабжения в момент возникновения спроса и его удовлетворение изменяет состояние всей системы снабжения (например, в благоприятном состоянии опорожнением продуктопровода можно покрывать другой актуальный спрос по содержимому трубопровода, в противном случае можно занимать те резервуары, которые моглы бы обеспечить опорожнение трубопровода в следующем цикле). Решение в связи с удовлетворением некоторого актуального спроса необходимо проанализировать и с той точки зрения, что новое состояние системы было благоприятным с точки зрения удовлетворения прогностизируемой последовательности спросов.

Следует обратить внимание на то, что в каждом случае, когда в момент возникновения спроса при i-том пункте потребления находится продукт, отличающий от требуемого j-того продукта, необходимо опорожнение участка между узлом и i-тым пунктом потребления, то есть продуктопровод занимается транспортировкой, которая не требуется в данный момент. Это явление снижает эффективную пропускную способность продуктопровода и даже может случиться, что некуда опорожнять соедржимое данного участка (возможные приемные резервуары полны) и при этом продуктопровод временно должен останавливаться.

Можно определить некоторые показатели, которые могут использоваться для качественной оценки системы и в эвристических алгоритмах решения задачи. Таким показателем является период дополнения.

$$t^{j}_{ip} = \frac{M^{j}_{i}}{z^{j}_{i}} \qquad [\text{час}]$$

то есть в случае потребления z^{j}_{i} к i-тому пункту потребления не позже через t^{j}_{ip} **час** следует снова поставлять j-той продукт (если при предыдущей поставке резервуар полностью наполнился и предусматривается его полное опорожнение).

Минимальный период **дополнения** $t^{p}_{мин} = \min\{t^{j}_{ip}\}$ есть существенной характеристикой системы, он является верхним пределом планируемого интервала последовательности поставок, если данный спрос следует удовлетворить по продуктопроводу.

Разброс периодов **дополнения** также характеризует систему снабжения, показывает "узкие места". В хорошо запроектированных системах периоды **дополнения** должны быть близкими, за исключением того случая, когда один из продуктов поставляется в значительно большем количестве и продуктопровод обычно наполнен этим продуктом. Для этого продукта периоды **дополнения** могут быть в принципе меньше (достаточно часто можно отобрать этот продукт из продуктопровода). К этому еще возвращаемся при анализе вопросов проектирования РСС.

Очевидно с технологической точки зрения, что периоды **дополнения** должны быть как можно больше, потому что "переключения" связаны с убытками из-за смешивания на разделе фаз и за счет уменьшения эффективной пропускной способности продуктопровода. Это проще всего достичь увеличением емкости резервуаров на пунктах потребления, но такое решение требует технико-экономического анализа.

Очень важным показателем является изменение периодов дополнения во времени. Если оно небольшое для нескольких интервалов планирования, то колебание потребления во времени (по отношению емкости резервуаров) является медленным возмущением, в противном случае - быстрым возмущением. Это влияет на стратегию управления, в первом случае следует стремиться к оптимизации управления на несколько периодов планирования (использовать, например, "скользящее" планирование), во втором случае следует стремиться к максимальному удовлетворению актуального спроса.

Отметим, что всегда наблюдается сезональное колебание периодов дополнения, поэтому задачу нельзя решать разработкой нескольких "расписаний" поставок.

Коэффициент нагрузки продуктопровода, который является частным необходимых поставок в заданном интервале времени на теоретическую пропускную способность продуктопровода:

$$\zeta = \frac{\sum_{i=1}^{n} \sum_{i=1}^{m} z_i^j}{w_{cs}} \cdot 100 \quad [\%]$$

также можно использовать для качественных оценок. Для конкретной РСС и при заданном ассортименте продуктов, анализируя предыдущие поставки, можно оценить тот диапазон этого коэффициента, в рамках которого задачу можно реализовать. Если коэффициент нагрузки, рассчитанный для актуальной поставки, меньше этого диапазона, то продуктопровод недогружен, если выше - он перегружен.

Отметим, что существуют тривиальные условия снабжения по продуктопроводу, которые целесообразно проверить перед началом планирования. К этим относятся такие условия,

как $w_i > z_i^j$ для всех $i=1,2,\ldots,n;$ $j=1,2,\ldots,m;$

то есть скорость поставки должна быть больше скорости потребления для всех пунктов потребления и продуктов.

4. СТРАТЕГИЯ ЭКСПЛУАТАЦИИ ПРОДУКТОПРОВОДА С ТОЧКИ ЗРЕНИЯ ТЕХНОЛОГИИ И МАТЕМАТИЧЕСКОЙ МОДЕЛИ

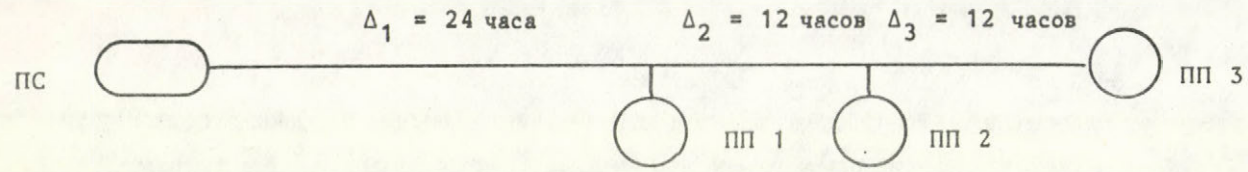
Рассмотрим протекание во времени процессов принятия и реализации решений в связи с эксплуатацией продуктопровода (рис.2). **Пусть известны параметры состояния системы** (запасы по пунктам потребления, содержимое продуктопровода) в момент t_m а также прогноз потребления на следующий интервал $T = \{t_0, t_v\}$. Момент принятия решения t_d . Как относятся друг к другу эти моменты времени, насколько раньше необходимо принимать решения по "расписанию" на следующий период?

Возможны два случая. В первом случае в момент t_d транспортировка по продуктопроводу не идет и в текущем интервале $T'=\{t_0', t_v'\}$ **поставка уже не предусмотрена. В таком случае** $t_m \leq t_d < t_0$. Определим те моменты, когда запасы на пунктах потребления кончаются:

$$\tau_i^j = t_m + \frac{q_i^j(t_m)}{z_i^j}$$

При этом должно выполняться $\min\{\tau_i^j\} \geq t_0$, в противном случае предыдущее планирование было ошибочным.

Нельзя забывать, что продуктопровод наполнен каким-то продуктом. Оптимальное опорожнение продуктопровода можно рассматривать отдельной задачей, в результате "распределения" содержимого трубопровода соответствующие моменты τ_i^j задвигаются во времени и получается корректированный вектор $\theta = \{\theta_i^j\}$.



Принципиальная схема продуктопровода для транспортировки трех продуктов. Указаны времени запаздывания (предполагая, что интенсивность отбора равна во всех ПП и одновременно отбирается только один продукт)
ПС - Пункт снабжения ПП - Пункт потребления

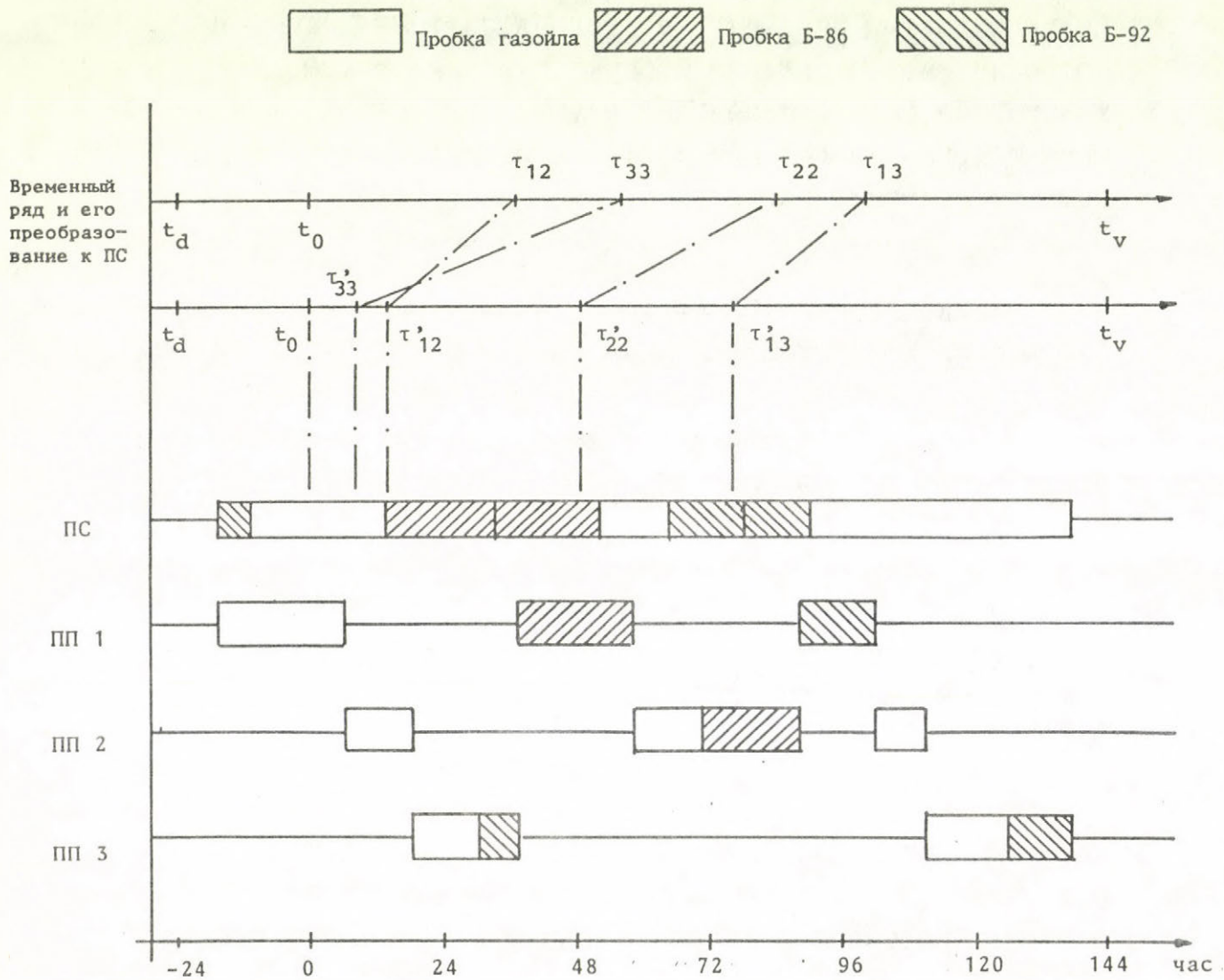


Рис. 2. Временные характеристики работы продуктопровода с указанием нагнетания и отбора пробок. В 6-ти дневном периоде планирования запасы кончаются в 4-х резервуарах. Видно, что задача имеет решение только при условии, что нагнетание продуктов начинается еще до момента t_0. Пробки газойла №№ 2 и 5 необходимы для наполнения соответствующих участков трубопровод

Если началом пуска транспортировки будет t_s, то продукт, иагнетанный в продуктопровод в момент t_s появляется на i-той пункте потребления запаздыванием Δ_i (зависит также от распределения содержимого продуктопровода). Для того, чтобы удовлетворить самый ранный спрос интервала планирования T, необходимо:

$$t_s + \Delta_i \leq \Theta_{min}$$

далее

$$t_d \leq t_s$$

$$\tau_{min} \leq \Theta_{min}$$

$$\tau_{min} \geq t_0$$

где τ_{min}, Θ_{min} - $\min \{\tau_i^j\}$, $\min \{\theta_i^j\}$, соответственно.

В самом неблагоприятном случае, когда

$$\tau_{min} = t_0$$

и в продуктопроводе находится отличающегося от требованного продукт, то есть

$$\tau_{min} = \Theta_{min}$$

позднейший срок начала транспортировки опережает начало интервала планирования

$$t_s^{max} = (t_0 - \Delta_i)$$

О позднейшем сроке начала транспортировки говорим потому, что если после наиболее ранним "заказом" следуют вторые, третие и т.п. заказы, может случиться, что они могут быть удовлетворены только тогда, если транспортировка первого продукта начинается раньше позднейшего срока, не дожидаясь полного опорожнения данного резервуара.

Если продуктопровод достаточно загружен и нет значительных перерывов работы, актуальный цикл транспортировки необходимо заканчивать таким образом, чтобы обеспечить по возможности "оптимальную ситуацию" системы для следующего интервала планирования. Для этого часто необходимо пересмотреть реше-

ния по последним поставкам. На основе изложенного можно сделать вывод о том, что требования по поставкам на следующий интервал должны проверяться заблаговременно, перед заканчиванием **текущего** интервала, но не позже вышепоказанного срока.

Важным **вопросом** является определение длины интервала планирования. Он конечно зависит с одной стороны от характеристик РСС, с другой стороны от скорости изменения потребления продуктов.

Если потребление продуктов изменяется быстро, то долгосрочные прогнозы являются очень сомнительными и надо стремиться к планированию на небольшие интервалы, часто повторяя планирование (при возникновении нового, обоснованного спроса).

Если потребление продуктов изменяется медленно, следует оптимизировать на длительный период и целесообразным является реализация некоторой периодической стратегии поставок. Вообще говоря, периодическая стратегия поставок требует выполнение различных условий для РСС. Таким образом, в начальном моменте имеется двойная задача: следует определить подходящую периодическую стратегию поставок и ее условий; и привести РСС в такое состояние, в котором можно начать периодические поставки. В качестве примера на рис. 3. дается временная диаграмма периодических поставок в несложной РСС.

Отметим, что в случае одних систем возможен большой набор стратегий удовлетворения всех требований путем периодических поставок, в случае других систем совершенно нельзя обеспечить периодическое снабжение. Возможность реализации периодической стратегии поставок является свойством РСС и при реконструкционных работ следует иметь в виду это характеристику. Конечно, возможно также реализация периодических поставок только в отдельных сезонах, в отсутствие экстренных ситуаций и т.д.

Диаграмма изменения запасов в резервуарах

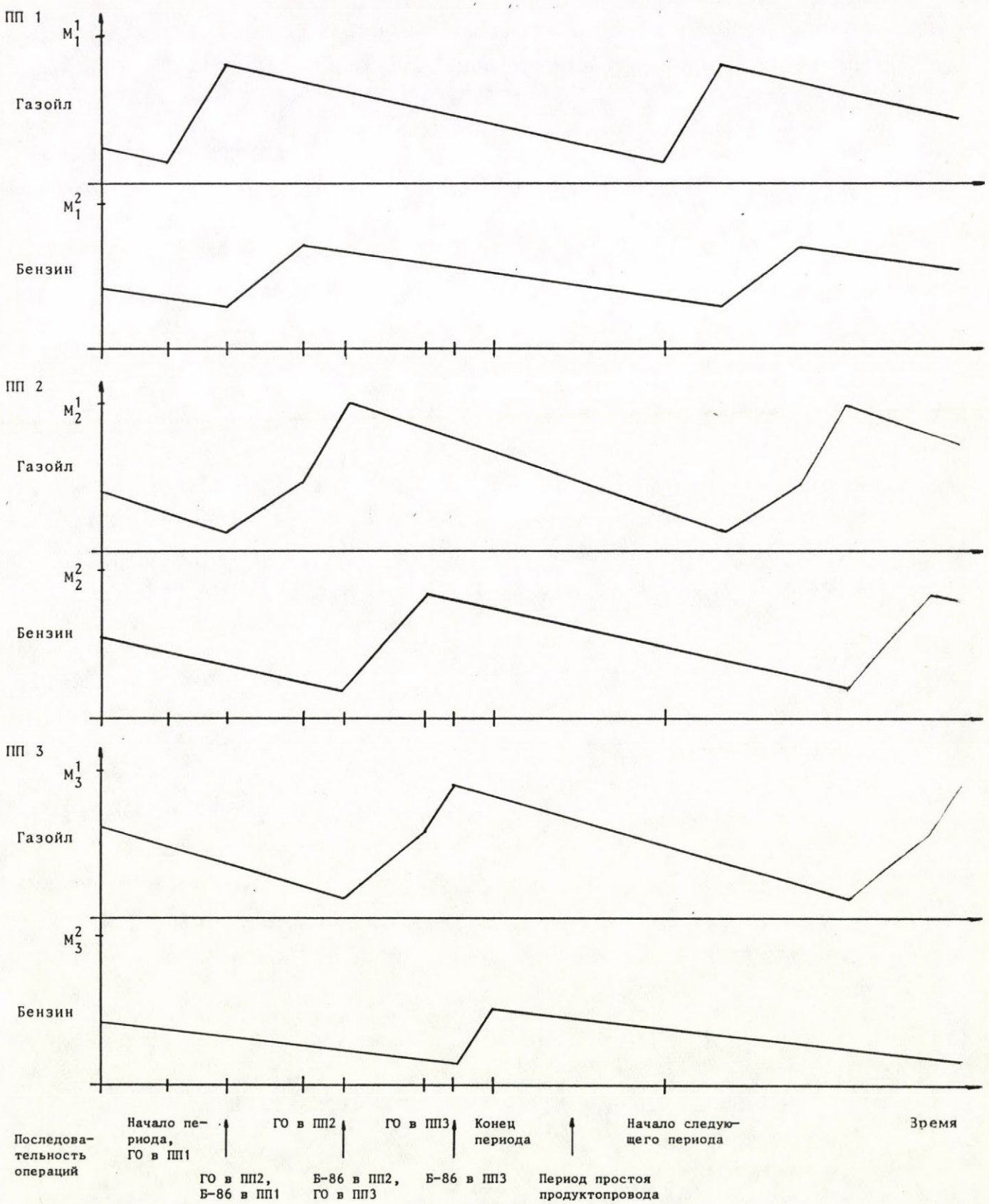


Рис. 3. Схема организации периодической поставки в случае двух продуктов (газойла - ГО и бензина - Б-86) в системе с тремя ПП. Для компенсации колебаний потребления можно варьировать операции отбора и использовать период простоя

Из изложенного уже видно, что в процессе управления продуктопроводом необходимо совместно решать дискретную задачу (определение последовательности "адресованных" пробок) с задачей непрерывного характера (определение объема, "длины" этих пробок).

Если РСС включает в себя n пунктов потребления и в системе транспортируются m продуктов, то количество теоретическим возможных последовательностей пробок - при определенных ограничениях - равно $(n \cdot m)!$, то есть в случае не очень сложных систем (например, 5 пунктов потребления, 3 продукта) уже получается очень большое число для возможных последовательностей. Проверка такого количества вариантов, при этом определяя оптимальный объем "пробок" продуктов, нереализуема.

По моему мнению, необходимо предварительно задавать (или выбирать эвристическим путем, исходя из технологических или других рассуждений) несколько реальных последовательностей поставок и возвращаться к дискретной задаче только тогда, если спрос не может быть обеспечен при таких последовательностях пробок.

Что касается непрерывной задачи, определения объема пробок, отметим, что имеем временной ряд Θ_i^j $(i=1,2,\ldots,n;\ j=1,2,\ldots,m)$, характеризующий израсходование запасов по пунктам потребления. Зная последовательность поставок, в зависимости от объема пробок и скорости их отбора на отдельных пунктах потребления можно рассчитать моменты времени t_i^j, когда фронт j-того продукта поступает к i-тому пункту потребления. При этом объем пробок должен определяться таким образом, что неравенства

$$t_i^j \leq \Theta_i^j$$

$$\tau_{ic}^j \geq t_v$$

где τ_{ic}^j - момент следующего опорожнения резервуара;

t_v - конец интервала планирования T;

выполнились для всех i и j.

Если эта задача имеет решение, то оно недалеко от оптимального, потому что основной компонент целевой функции, транспортные расходы, является минимальным. Если задача не имеет решения, можно проверить другие возможные последовательности и можно сопоставить различные варианты по целевой функции, предполагая, что "недостающие" продукты будут поставляться по независимому пути транспортировки.

В задачу необходимо включить далее набор условий, характеризующих возможность реализации полученного плана (опорожнение содержимого трубопровода и т.д.) или необходимо с помощью имитационной модели проверить реализуемость плана.

5. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ МОЩНОСТЕЙ ПУНКТОВ ПОТРЕБЛЕНИЯ, ПОДКЛЮЧЕННЫХ К ПРОДУКТОПРОВОДУ

В предыдущей части неоднократно отметили, что при наличии продуктопровода вся РСС является одной целой и любое изменение влияет на характеристики системы. Здесь желаем обратить внимание на отдельные факты, которые требуют нового подхода при проектировании мощностей пунктов потребления, подключенных к продуктопроводу.

На пунктах потребления емкости резервуаров более-менее пропорциональны обороту данного продукта. При наличии продуктопровода обычно хотя бы один из продуктов поставляется в количестве, значительно больше количества других продуктов (в ВНР таким продуктом является газойл). Это означает, что в большей части времени трубопровод заполнен газойлом и спрос по нему можно удовлетворить немедленно. Необходимые емкости для газойла должны определяться не по обороту этого продукта, а исходя из двух факторов:

- запасы на пунктах потребления должны быть достаточными на время поставки других продуктов (средний коэффициент занятости продуктопровода на транспортировку других продуктов можно рассчитать по сезонам);
- емкости резервуаров должны обеспечить возможность опорожнения газойла из продуктопровода.

С другой стороны, целесообразно иметь большие емкости для продуктов с меньшим потреблением (например, высокооктановый бензин), этим увеличить $t^{P}_{мин}$ - минимальный период пополнения. Как уже показывали, этим увеличивается общая (полезная) пропускная способность системы и снижаются потери.

Опыт показывает, что "узкие места" в системе с низким $t^{P}_{мин}$ усложняют задачу управления продуктопроводом и снижают экономичность эксплуатации, к ним часто необходимо или проще поставлять продукт по железной дороге или автотранспортом.

Соблюдение указанных требований не всегда требует сооружение новых резервуаров, часто достаточно по-новому распределить имеющиеся резервуары между продуктами. Этот факт также подтверждает целесообразность всестороннего анализа уже имеющихся систем с помощью имитационной модели РСС.

REGIONÁLIS BENZIN- ÉS GÁZOLAJ ELLÁTÓ RENDSZEREK SAJÁTOSSÁGAI TÖBB TERMÉK SZÁLLÍTÁSÁRA SZOLGÁLÓ TERMÉKVEZETÉK ESETÉN

Inzelt Péter

Összefoglaló

A termékvezeték - a többi szállítási móddal összehasonlítva - a motorbenzinek és gázolaj rendkívűl olcsó szállítását biztosítja, optimális kihasználása jelentős megtakarítást eredményez. A termékek váltakozva, "dugók" formájában haladnak a termékvezetékben. A feladat a "dugók" sorrendjének és nagyságának meghatározása olymódon, hogy - egyéb technológiai korlátok mellett - egyetlen tárolótéren sem fogyhat el egyetlen termék sem, de a tartályok sem tölthetők túl.

A termékvezeték szállítókapacítása függ az egyes igények sorrendjétől és nagyságától, tágabb értelemben a teljes ellátási rendszer állapotától. Minden igény kielégítése a teljes rendszer állapotát változtatja meg. Ebből a tényből következik az is, hogy termékvezeték üzemeltetése esetén a csatlakozó tárolóterek tartálykapacításait az ellátási rendszer tulajdonságaiból kiindulva kell meghatározni.

ON THE SPECIFIC ROLE OF A MULTIPRODUCT-PIPE-LINE IN THE REGIONAL PETROL- AND GASOLINE DISTRIBUTION SYSTEMS

Peter Inzelt

Summary

Pipelines give the opportunity of very low-cost transportation of light crude-oil products to compare them with the railway or road transport. The petrol and gasoline partions are moving sequentially as a set of "plungers". The sequence of "plungers" and their "length" (volume) should be defined so that - among other technological constraints - no product stock should be exhausted at the tank parks connected to the pipeline and over-charging the tanks is impossible, too.

The pipeline throughtput depends on the sequence and volume of product-demands, in general sense, on the state of the whole regional system. The satisfying of each demand changes the whole system's state. To achieve the optimal conditions for the multiproduct-pipeline operation, the tank-capacities of the tank-parks connected to the pipeline should be defined on the basis of characteristics of the whole distribution system.

MTA SZTAKI Közlemények 35/1986 p. 31-46

SOME RESULTS ABOUT THE STRUCTURE OF MINIMUM COVERS IN THE RELATIONAL DATABASE MODEL

TRAN THAI SON - VINH THI NGOC THANH

Institute of Informatics and Cybernetics of Hanoi

HO THUAN

Computer and Automation Institute, Hungarian Academy of Sciences

1. INTRODUCTION

In [2] the notion of equivalence classes of left sides $E_F(x)$ has been introduced by D. Maier, and it is shown that for any equivalent minimum sets of FDs F_1 and F_2, $|E_{F_1}(X)| = |E_{F_2}(X)|$ for any X.

D. Maier also proved that for each FD $X_i \to \bar{X}_i \in E_{F_1}(X)$ there exists a unique $Y_i \to \bar{Y}_i \in E_{F_2}(X)$ such that $X_i \overset{\bullet}{\longleftrightarrow} Y_i$. Therefore Y_i (resp. X_i) can be substitued for X_i (resp. Y_i) without changing the closure of F_1 (resp. F_2) i.e.

$$[\{F_1 \setminus (X_i \to \bar{X}_i)\} \cup (Y_i \to \bar{X}_i)] \equiv F_1$$

and

$$[\{F_2 \setminus (Y_i \to \bar{Y}_i)\} \cup (X_i \to \bar{Y}_i)] \equiv F_2.$$

So, the structure of left sides of FDs in minimum covers has been enough well discovered.
In this paper we investigate the structure of right sides of FDs in equivalent minimum covers, and try to find a certain sort of transformations for right sides.

In studying the structure of right sides of FDs, in minimum covers by the results of D. Maier just mentioned above, we can assume that *all equivalent minimum covers have the same set of left sides.*

2. DEFINITIONS AND NOTATION

In this section we recall some notions and results which are needed in the sequel. Let F be a set of functional dependencies (FD) defined over the universe of attributes

$$\Omega = \{A_1, A_2, \ldots, A_n\},$$

$$F = \{L_i \rightarrow R_i \mid L_i, R_i \subseteq \Omega, \quad i = 1,2,\ldots,m\}.$$

The closure F^+ of F is the set of all FDs which can be derived from F through Armstrong's inference rules [4,5].

Definition 2.1

A set of FDs F is *minimum* if there is no set G with fewer FDs than F such that $G \equiv F$ (i.e. $G^+ = F^+$).

Definition 2.2

Given a set of FDs F with $X \rightarrow Y$ in F^+. X directly determines Y under F, written $X \stackrel{\cdot}{\rightarrow} Y$, if $X \rightarrow Y \in [F \setminus E_F(X)]^+$, where $E_F(X)$ is the set of all FDs in F with left sides equivalent to X.

That is, no FDs with left sides equivalent to X are used to derive $X \rightarrow Y$.

Definition 2.3

A set of FDs F is *optimal* if there is no set of FDs G with fewer attribute symbols such that $G \equiv F$.
(Repeated symbols are counted as many times as they occur).

We have the following lemma and theorem:

Lemma 2.1 [6]

Let $V \to W$ be an FD in F^+ and let $X \to Y$ be an FD in F that participates in the Armstrong's derivation sequence for $V \to W$.

Then we have

$$V \to X, \quad VY \to W \in (F \setminus \{X \to Y\})^+$$

Theorem 2.1 [6]

If F_1, F_2 are nonredundant and equivalent sets of FDs, then

$$F_1 \equiv F_2 \equiv \{F_1 \setminus E_{F_1}(X)\} \cup E_{F_2}(X) \equiv \{F_2 \setminus E_{F_2}(X)\} \cup E_{F_1}(X).$$

Finally, as in [5] let us denote

$$R = \bigcup_{i=1}^{m} L_i \quad \text{and} \quad L = \bigcup_{i=1}^{m} L_i.$$

3. STRUCTURE OF MINIMUM COVERS

Denote by $LE_F(X)$ and $RE_F(X)$ the sets of attributes in left and right sides of FDs in $E_F(X)$ respectively, and instead of $[F \setminus \{X \to \bar{X}\}] \cup \{X \to \bar{X} \setminus Z_o\}$, sometimes for sake of simplicity, we write $F \setminus \{X \to Z_o\}$ if $Z_o \subseteq \bar{X}$ and $X \to \bar{X} \in F$.

We begin with the following fundamental theorem.

Theorem 3.1

Let F_1 and F_2 be two equivalent minimum sets of FDs, $X_1 \rightarrow \bar{X}_1 \in E_{F_1}(X)$, $Z_o \subseteq \bar{X}_1$ and $Z_o \cap RE_{F_2}(X) = \emptyset$.

Then there exists Z such that

$$X_1 Z \leftrightarrow X_1 Z_o \in [F_1 \backslash \{X_1 \rightarrow Z_o\}]^+.$$

Z_o and Z are said to be equivalent *via* X_1.

Proof.

Since $X_1 \rightarrow \bar{X}_1 \in E_{F_1}(X)$, there exists

$$X_1 \rightarrow \bar{Y}_1 \in E_{F_2}(X).$$

First of all, we show that $X_1 \rightarrow \bar{X}_1$ must participate in the Armstrong's derivation sequence for $X_1 \rightarrow \bar{Y}_1$ and vice versa. Assume the contrary that $X_1 \rightarrow \bar{X}_1$ does not participate in the derivation sequence for $X_1 \rightarrow \bar{Y}_1$.

So $X_1 \rightarrow \bar{Y}_1 \in [F_1 \backslash X_1 \rightarrow \bar{X}_1]^+$. Since F_1 and F_2 are non-redundant, $X_1 \rightarrow \bar{Y}_1$ cannot be derived from $F_2 \backslash E_{F_2}(X)$. On the other hand, by Theorem 2.1,

$F_2 \equiv F_1 \equiv \{F_1 \backslash E_{F_1}(X)\} \cup E_{F_2}(X) \equiv \{F_2 \backslash E_{F_2}(X)\} \cup E_{F_1}(X)$, it follows that there exists $X_h \rightarrow \bar{X}_h \in E_{F_1}(X)$ that participates in the derivation sequence for $X_1 \rightarrow \bar{Y}_1$.

By lemma 2.1 we have

$$X_1 \rightarrow X_h \in [F_1 \{X_1 \rightarrow \bar{X}_1, X_h \rightarrow \bar{X}_h\}]^+.$$

But, in that case, we have

$$(F_1 \backslash \{X_1 \rightarrow \bar{X}_1, X_h \rightarrow \bar{X}_h\}) \cup \{X_h \quad \bar{X}_1 \bar{X}_h\} \equiv F_1$$

which contradicts the fact that F_1 is minimum.
Thus $X_1 \rightarrow \bar{X}_1$ must participate in the Armstrong's sequence for $X_1 \rightarrow \bar{Y}_1$, and in turn, $X_1 \rightarrow \bar{Y}_1$ must participate in the Armstrong's derivation sequence for $X \rightarrow \bar{X}_1$. By Lemma 2.1, we have:

$$X_1 \bar{X}_1 \rightarrow \bar{Y}_1 \in \{F_1 \setminus (X_1 \rightarrow \bar{X}_1)\}^+$$

and

$$X_1 \bar{Y}_1 \rightarrow \bar{X}_1 \in \{F_2 \setminus (X_1 \rightarrow \bar{Y}_1)\}^+.$$

Now we can split $X_1 \rightarrow \bar{X}_1$ into $X_1 \rightarrow Z_o$ and $X_1 \rightarrow X_1 \setminus Z_o$.

If $X_1 \rightarrow Z_o$ does not particpiate in the Armstrong's derivation sequences for $X_1 \bar{Y}_1 \rightarrow \bar{X}_1$ then $\bar{Y}_1$ is the required Z.
Indeed, in that case we have

$$X_1 \bar{Y}_1 \rightarrow Z_o \in \llbracket \{F_1 \setminus (X_1 \rightarrow \bar{X}_1)\} \cup \{X_1 \rightarrow (\bar{X}_1 \setminus Z_o)\} \rrbracket^+.$$

Moreover,

$$X_1 \bar{X}_1 \rightarrow \bar{Y}_1 \in \{F_1 \setminus (X_1 \rightarrow \bar{X}_1)\}^+ \subseteq$$

$$\subseteq \llbracket \{F_1 \setminus (X_1 \rightarrow \bar{X}_1)\} \cup \{X_1 \rightarrow (\bar{X}_1 \setminus Z_o)\} \rrbracket^+,$$

$$X_1 \rightarrow (\bar{X}_1 \setminus Z_o) \in \llbracket \{F_1 \setminus (X_1 \rightarrow \bar{X}_1)\} \cup X_1 \rightarrow \bar{X}_1 \setminus Z_o \rrbracket$$

and $X_1 Z_o \rightarrow X_1 \bar{X}_1$ can be derived from $X_1 \rightarrow \bar{X}_1 \setminus Z_o$, so using the transitivity rule, we get:

$$X_1 Z_o \rightarrow \bar{Y}_1 \in \llbracket \{F_1 \setminus (X_1 \rightarrow \bar{X}_1)\} \cup \{X_1 \rightarrow \bar{X}_1 \setminus Z_o\} \rrbracket^+$$

Now consider the case where $X_1 \rightarrow Z_o$ participates in the

derivation sequence in F_1 for $X_1\ \bar{Y}_1 \rightarrow \bar{X}_1$.

Since $X_1\ \bar{Y}_1 \rightarrow \bar{X}_1 \in F_2^+$, it can be derived from FDs in F_2, which in turn can be derived from FDs in F_1.

So there exists at least on FD $X_2 \rightarrow \bar{Y}_2$ in F_2 such that $X_2 \rightarrow \bar{Y}_2$ participates in the derivation sequence in $F_2 \backslash (X_1 \rightarrow \bar{Y}_1)$ for $X_1\ \bar{Y}_1 \rightarrow \bar{X}_1$, and $X_1 \rightarrow Z_o$ will participate in the Armstrong's derivation sequence in F_1 for $X_2 \rightarrow \bar{Y}_2$.

By virtue of lemma 2.1, we have:

$$x_1\ \bar{Y}_1 \rightarrow X_2 \quad \text{and} \quad X_2 \rightarrow X_1 \in F_1^+.$$

And from $X_1 \rightarrow \bar{Y}_1 \in F_2 \subseteq F_1^+$, we conclude that: $X_2 \leftrightarrow X_1$.

So $(X_2 \rightarrow \bar{Y}_2) \in E_{F_2}(X)$. By the hypothesis of the theorem,

$$Z_o \cap R\ E_{F_2}(X) = \emptyset.$$

It follows that

$$X_2 \rightarrow \bar{Y}_2 \neq X_1\ \bar{Y}_1 \rightarrow \bar{X}_1.$$

Moreover, by lemma 2.1, we have:

$$X_1\ \bar{Y}_1\ \bar{Y}_2 \rightarrow Z_o \in [F_2 \backslash \{X_1 \rightarrow \bar{Y}_1,\ X_2 \rightarrow \bar{Y}_2\}]^+$$

Two cases can be happen:

If $X_1 \rightarrow Z_o$ does not participate in the derivation sequence for $X_1\ \bar{Y}_1\ \bar{Y}_2 \rightarrow Z_o$ in F_1, we prove that $\bar{Y}_1\ \bar{Y}_2$ is the Z to be found.

Indeed, we have $X_1\ \bar{Y}_1\ \bar{Y}_2 \rightarrow Z_o \in [F_1 \backslash (X_1 \rightarrow Z_o)]^+$

Moreover, using an argument similar to the one given at the beginning of the proof, we can prove that $X_1 \rightarrow \bar{X}_1$ must

participate in the Armstrong's sequence for $X_1 \rightarrow \bar{Y}_j$, $j = 1,2,\ldots,k$ (k is the cardinality of $E_{F_1}(X)$). So, from

$$X_1 Z_o \rightarrow \bar{Y}_i \in [F_1 \setminus (X_1 \rightarrow Z_o)]^+, \quad i = 1,2, \quad \text{we have}$$

$$X_1 Z_o \rightarrow \bar{Y}_1 \bar{Y}_2 \in [F_1 \setminus (X_1 \rightarrow Z_o)]^+.$$

Now consider the case where $X_1 \rightarrow Z_o$ participates in the Armstrong's derivation sequence in F_1 for $X_1 \bar{Y}_1 \bar{Y}_2 \rightarrow Z_o$.

Similarly as before, there exists

$$X_3 \rightarrow \bar{Y}_3 \in [E_{F_2}(X) \setminus \{X_1 \rightarrow \bar{Y}_1, X_2 \rightarrow \bar{Y}_2\}].$$

Such that $X_3 \rightarrow \bar{Y}_3$ participates in the Armstrong's derivation sequence for $X_1 \bar{Y}_1 \bar{Y}_2 \rightarrow Z_o$.
The process continues. But as $|E_{F_2}(X)| \neq +\infty$ so it must be finished at step h, where $h \leq |E_{F_2}(X)|$.

At that moment, we have

$$X_1 (\bar{Y}_1 \bar{Y}_2 \ldots \bar{Y}_h) \rightarrow Z_o,$$

and $$X_1 Z_o \rightarrow (\bar{Y}_1 \ldots \bar{Y}_h) \in [F_1 \setminus \{X_1 \rightarrow Z_o\}]^+$$

The proof is complete.

The theorem shows that attributes of the right side of an FD $X_i \rightarrow \bar{X}_i$ belonging to $E_F(X)$ in a minimum cover F can be divided in two classes. The first class consists of invariant attributes, that is, they must appear in right sides of FDs in every $E_{F_j}(x)$, where F_j is any minimum cover equivalent to F. The second class consists of all attributes that belong to a set $Z_o \subseteq \bar{X}_i$, where $X_i \rightarrow \bar{X}_i \in E_F(X)$ such that $\exists Z \neq Z_o$ and $X_i Z_o \leftrightarrow X_i Z \in [\{F \setminus (X_i \rightarrow Z_o)\}]^+$.

For attributes of the second class, we have:

Corollary 3.1

With Z_o and Z as determined in Theorem 3.1, we can replace Z_o in $(X_1 \rightarrow \bar{X}_1) \in F$ by Z and so doing, obtain an equivalent minimum cover.

Proof.

The proof is straight-forward.

From $X_1 \rightarrow Z_o \in F$ and $X_1 Z_o \rightarrow Z \in [F \backslash X_1 \rightarrow Z_o]$ we obtain $X_1 \rightarrow Z$ and conversely, from $X_1 \rightarrow Z$ and $X_1 Z \rightarrow Z_o \in (F \backslash \{X_1 \rightarrow Z_o\})^+$ we obtain $X_1 \rightarrow Z_o$.

Remark 3.1 From Theorem 3.1 and Corollary 3.1, we found the transformation rules for right sides of FDs in equivalent minimum covers as follows:
First, there are attributes of right sides that can be replaced by equivalent sets *via* left sides. Such transformations can be done in a single FD too.
Second, there are attributes of right sides that are invariant (i.e. always present) and only change places in right sides of the equivalence class.
In that case, transformation must be done simultaneously in several FDs of the equivalence class.

Corollary 3.2 Let $Z_o \subseteq \bar{X}_1$ where $X \rightarrow \bar{X}_1 \in E_{F_1}(X)$ and $Z_o \subseteq R \backslash L$. *)

Then in any minimum cover $F_2 \equiv F_1$, we have

$$Z_o \subseteq R\ E_{F_2}(X).$$

*) *Here we assume that all minimum covers are in natural reduced form and are left-reduced. In that case $R \backslash L$ is the same for all minimum equivalent covers* [7]

Proof First observe that if there exists Z such that $(X_1\ Z_o \rightarrow Z) \in F^+$, then there exists an FD $(Y \rightarrow W) \in F$ such that $Y \cap Z_o \neq \emptyset$.
Thus

$$Z_o \not\subseteq R \setminus L$$

Therefore *all attributes in* $R\setminus L$ *belong to the first class*, i.e., invariant in equivalence classes of equivalent minimum covers.

Let $E_F(X)$ is the set of following FD_s:

$$\begin{aligned}
&X_1 \rightarrow \tilde{X}_1\ \tilde{X}'_1\\
&X_2 \rightarrow \tilde{X}_2\ \tilde{X}'_2\\
&\cdots\cdots\cdots\cdots\\
&X_k \rightarrow \tilde{X}_k\ \tilde{X}'_k\ ,
\end{aligned}$$

where $\tilde{X}_i \subseteq LE_F(X)$ and $\tilde{X}'_i \cap LE_F(X) = \emptyset$. We can split $E_F(X)$ into

$$\begin{aligned}
&X_1 \rightarrow \tilde{X}_1\\
&X_2 \rightarrow \tilde{X}_2\\
&\cdots\cdots\cdots\\
&X_k \rightarrow \tilde{X}_k\\
&X_1 \rightarrow \tilde{X}'_1\\
&X_2 \rightarrow \tilde{X}'_2\\
&\cdots\cdots\cdots\\
&X_k \rightarrow \tilde{X}'_k\ .
\end{aligned}$$

Consider the first k FDs. They can be replaced by the following FDs while not altering the closure.

$$X_{i_1} \rightarrow X_{i_2}$$

$$X_{i_2} \rightarrow X_{i_3}$$

$$\vdots$$

$$X_{i_{k-1}} \rightarrow X_{i_k}$$

$$X_{i_k} \rightarrow X_{i_1}$$

where $(i_1, i_2, \ldots, i_k)$ is a permutation of $(1,2,\ldots,k)$.

Let A be any attribute in $R\ E_F(X)$. Then, there exists i such that $A \in \tilde{X}_i\ \tilde{X}'_i$. If $A \in \tilde{X}_i \subseteq LE_F(X)$, then there exists j such that $A \in X_j$.

With any p, $1 \leq p \leq k$, $p \neq j$, we can construct a new cover equivalent to F by replacing $E_F(X)$ by:

$$X_p \rightarrow X_j\ \tilde{X}'_p$$

$$X_j \rightarrow X_{i_3}\ X'_j$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$X_{i_{k-1}} \rightarrow X_{i_k}\ \tilde{X}'_{i_{k-1}}$$

$$X_{i_k} \rightarrow X_p\ \tilde{X}'_{i_k}$$

where $(p, j, i_3, i_4, \ldots, i_k)$ is a permutation of $(1,2,\ldots,k)$.

After reducing right sides, if A is an invariant attribute, A must belong to the right side of the FD that has p as index.

If $A \in \tilde{X}'_i \not\subseteq LE_F(X)$ then with any p, $1 \leq p \leq k$, we can construct a new minimum cover equivalent to F by replacing $E_F(x)$ by:

$$X_1 \rightarrow X_2 \; \tilde{X}'_2$$

$$X_2 \rightarrow X_3 \; \tilde{X}'_3$$

$$\cdots\cdots\cdots\cdots$$

$$X_{i-1} \rightarrow X_i \; \tilde{X}'_p$$

$$\cdots\cdots\cdots\cdots$$

$$X_p \rightarrow X_{p+1} \; \tilde{X}'_i$$

$$\cdots\cdots\cdots\cdots$$

$$X_k \rightarrow X_1 \; \tilde{X}'_1$$

After reducing right sides, if A is an invariant attribute, A must belong to the right side of the FD in $E_F(X)$ that has p as index.

For these reasons, we say that invariant attributes in right sides can be distributed enough freely in right sides of FD_s in $E_F(X)$.

However, FDs in an equivalence class must satisfy the following property.

Property 3.1 Let $E_F(X) = \{X_j \rightarrow \bar{X}_j \mid j = \overline{1,k}\}$.

Then $\forall i \; \exists j$ such that

$$X_i \rightarrow X_j \in [\{F \setminus E_F(X)\} \cup \{X_i \rightarrow \bar{X}_i\}]^+$$

Proof Let be given arbitrary i,p. Since $X_i \rightarrow X_p$ is in F^+, we have

$$X_i \supseteq Z_{h_1}$$

$$X_i \, W_{h_1} \supseteq Z_{h_2}$$

$$\cdots\cdots\cdots\cdots\cdots$$

$$X_i \, W_{h_1} \cdots W_{h_q} \supseteq X_p$$

where $(Z_{h_t} \rightarrow W_{h_t}) \in F$, $t = \overline{1,q}$.

Choose $Z_{h_\ell} \rightarrow W_{h_\ell} \in E_F(X)$ with ℓ as small as possible.

With such Z_{h_ℓ}, we have the required result. In the case, such Z_{h_ℓ} does not exist, X_p is the required X_j.

Thus, in spite of the relative arbitrary distribution, each right side of an FD in the equivalence class must carry enough information such that, together with FDs in $F \setminus E_F(X)$, an FD of the form $(X_i \rightarrow X_j) \in F^+$ can be derived, ensuring the equivalence of left sides.

4. "QUASI OPTIMAL" COVER

Using the results just mentioned in 3. we can introduce the notion of "quasi optimal" cover. In [2] Marier defined the optimal cover (see Def. 2.3) and shown that the optimal cover problem is NP-complete.

However from the point of view of effective memory management this does not mean that there is no problem to be discussed, even in the case this optimal cover is found.

Consider k FDs of $E_F(X)$ in a minimum cover.

$$E_F(X) = \{X_i \rightarrow \tilde{X}_i \; \tilde{X}'_i \mid i = \overline{1,k}\}.$$

If we replace $E_F(X)$ by

$$\{X_1 \rightarrow X_2,\ X_2 \rightarrow X_3, \ldots, X_k \rightarrow X_1,\ X_1 \rightarrow \bigcup_{i=1}^{k} \tilde{X}'_i\},$$

then, for k first FDs we have only to take care of (to manage) their left sides, in opposite of the case of $E_F(X)$ we must manage both of left and right sides of all its FDs.

Moreover with $Z_o \subseteq \bigcup_{i=1}^{k} X'_i$, if there exists $Z \subseteq LE_F$ such that

$$X_i \; Z_o \leftrightarrow X_i \; Z \in [F\backslash\{X_i \rightarrow Z_o\}]^+,$$

then we can replace $X_1 \rightarrow \bigcup_{i=1}^{k} \tilde{X}'_i$ by the FD:

$X_1 \rightarrow \bigcup_{i=1}^{k} \tilde{X}'_i \backslash Z_o$ and still obtain an equivalent cover.

We close our paper with an algorithm to find the "quasi optimal" cover, in the above sense:

Input: The set G of FDs.

Output The "quasi optimal" cover F with $F^+ = G^+$

Method

Step 1 From G, find the minimum cover

$$F_1 = \text{MINIMIZE}(G) \quad [\text{see } 2].$$

We obtain the equivalence classes $E_{F_1}(X^1), \ldots, E_{F_1}(X^s)$ with the corresponding sets of FDs.

$$\{X_i^1 \rightarrow \bar{X}_i^1 \mid i = \overline{1,k_1}\}, \ldots, \{X_i^s \rightarrow \bar{X}_i^s \mid i = \overline{1,k_s}\}.$$

Step 2 For each equivalence class $E_{F_1}(X^\ell)$, $\ell = 1,2,\ldots,s$

Set $M_\ell = RE_{F_1}(X^\ell)\backslash L^{(1)}$ *).

---

*) $L^{(1)}$ *is the union of all left sides of FDs in* F_1.

For $i = \overline{1,k_\ell}$, consider $X_i^\ell \to \bar{X}_i^\ell$.

$\forall A \in \bar{X}_i^\ell$,

If $A \in LE_{F_1}(X^\ell)$ we omit it,

If $A \bar{\in} LE_{F_1}(X^\ell)$ then

for each $Z = LE_{F_1}(X^\ell) \backslash X_i^\ell$ check whether

$(X_i^\ell\ Z \to A) \in [F \backslash \{X_i^\ell \to A\}]^+$?

if true then we omit it

if false them $M_\ell = M_\ell \cup \{A\}$.

Step 3 At the end of Shep 2 we obtain M_ℓ for each equivalence class, and F is the following cover:

$$F = \bigcup_{\ell=1}^{s} \{X_1^\ell \to X_2^\ell,\ X_2^i \to X_3^\ell, \ldots, X_{k_\ell}^\ell \to X_1^\ell, X_1^\ell \to M_\ell\}$$

REFERENCES

[1] Bernstein, P.A., Synthesizing third normal form relations from functional dependencies. ACM Trans. Data base Syst. 1, 4 (Dec. 1976), 277-298.

[2] Maier, D., Minimum covers in the relational data base model. J. ACM 27 (Oct. 1980), 664-674.

[3] Ausiell, G. et al: Graphs algorithms for functional dependency manipulation. J. ACM 30 (Oct. 1983), 752-766.

[4] Ullman, J.D., Principles of Data base Systems. 2nd ed. Computer Science Press. Potomac, Md., 1982.

[5] Ho Thuan and Le van Bao, Some results about keys of relational schemas. Acta Cybernetica, Tom 7, Fasc. 1, Szeged 1985, pp. 99-113.

[6] Ho Thuan, Direct determination and FD-graph, MTA SZTAKI Közlemények 34/1986.

[7] Ho Thuan, Some invariants of covers for functional dependencies, MTA SZTAKI Közlemények 34/1986

Néhány eredmény a relációs adat-bázis modellben lévő minimális lefedések strukturájára vonatkozóan

Tran Thai Son, Dinh Thi Ngoc Thanh, Ho Thuan

Összefoglaló

A nem-redundáns és minimális lefedéseket az [1], [2], [3]-ban vizsgálták meg és használták az adat-bázisok logikai tervezésében. E cikkben a minimális lefedésekben levő funkcionális függőségek jobb oldalainak strukturáját vizsgáljuk. Egy algoritmust javasolunk, amely a "kvázi-optimális" lefedést találja meg.

Несколько результатов о структуре минимальных покрытий в реляционной модели баз данных

Тран Тхаи Сон, Динх Тхи Нгоц Тханх, Хо Тхуан

Р е з ю м е

В [1], [2], [3] изучались минимальные и нон-редундантные покрытия и использовали их в логическом проектировании баз данных. В этой статье мы изучаем структуру правых сторон функциональных зависимостей в минимальных покрытиях. Даем алгоритм для нахождения "квази-оптимальных" покрытий.

MTA SZTAKI Közlemények 35/1986 p. 47-52

SOME REMARKS ON THE ALGORITHM OF LUCCHESI AND OSBORN

HO THUAN

Computer and Automation Institute, Hungarian Academy of Sciences

Let $S = \langle \Omega, F \rangle$ be a relation scheme where $\Omega = \{A_1, A_2, \ldots, A_n\}$ the universe of attributes, and $F = \{L_i \rightarrow R_i \mid L_i, R_i \subseteq \Omega, \quad i = 1,2,\ldots,m\}$ - the set of functional dependencies.

In [2] C.L. Lucchesi and S.L. Osborn provided a very interesting algorithm to determine the set of all keys for any relation scheme $S = \langle \Omega, F \rangle$.

The algorithm has time complexity

$$O(|F| \; |K_S| \; |\Omega| (|K_S| + |\Omega|)),$$

following our notation, i.e. in time polynomial in $|\Omega|, |F|$ and $|K_S|$,
where

$|F|$ - the cardinality of F, and

K_S - the set of all keys for S.

We reproduce here this algorithm with some modifications in accordance to our notation.

Algorithm OL1. Set of all keys for $S = \langle \Omega, F \rangle$

Comment K_S is the set of keys being accumulated in a sequence which can be scanned in the order in which the keys are entered;

```
K_S ← {key*)(Ω,F,Ω)};
for each K in K_S do
   for each FD (L_i → R_i) in F do
      T ← L_i ∪ (K\R_i);
      test ← true;
      for each J in K_S do
        if T includes J then test ← false;
      if test then K_S ← K_S ∪ {key(Ω,F,T)}
   end
end;
return K_S.
```

The following simple remarks in some cases can be used to improve the performance of the algorithm of Lucchesi and Osborn.

Remark 1 To find the first key for $S = \langle \Omega, F \rangle$, instead of Ω, it is better to use the superkey $(\Omega \setminus R) \cup (L \cap R)$ and algorithm 1 in [1], and instead of the algorithm key (Ω,F,T), it is better to use algorithm 2 [1] for finding one key for S included in a given superkey T.

Remark 2 In [1] we have proved that

$$R \setminus L \subseteq \Omega \setminus H, \quad H = \bigcup_{K \in K_S} K$$

i.e., $R \setminus L$ consists only of non-prime attributes. Therefore if $R_i \subseteq R \setminus L$ then

$$R_i \cap K = \emptyset, \quad \forall K \in K_S.$$

and $$L_i \cup (K \setminus R_i) \supseteq K.$$

*) key (Ω,F,X) is the algorithm which determines a key for $S = \langle \Omega, F \rangle$ that is a subset of a specified superkey X.

That means, when computing $T = L_i \cup (K \setminus R_i)$, we can neglect all FDS $L_i \to R_i$ with $R_i \subseteq R \setminus L$, for every $K \in K_S$.

Let us denote

$$\bar{F} = F \setminus \{L_j \to R_j \mid L_j \to R_j \in F \text{ and } R_j \subseteq R \setminus L\}$$

Remark 3 With a fixed K in K_S, it is clear that if $K \cap R_i = \emptyset$ then $L_i \cup (K \setminus R_i) \supseteq K$. In that case it is not necessary to continue to check whether T includes J for each J in K_S.
So, it is better to compute T by the following order:

$$T = (K \setminus R_i) \cup L_i .$$

Remark 4. The algorithm of Lucchesi and Osborn is particularly effective when the number of keys for $S = \langle \Omega, F \rangle$ is small. But basing on what information one can conclude that the number of keys for $S = \langle \Omega, F \rangle$ is small? There is no general answer for all the cases, and it is shown in [3] that the number of keys for a relation scheme $S = \langle \Omega, F \rangle$ can be factorial in $|F|$ or exponential in $|\Omega|$, and that both of these upper bounds are attainable.
However, it is shown (in [1], Corollary 1) that

$$|K_S| \leq C_h^{\lceil h/2 \rceil}$$

where h is the cardinality of $L \cap R$.
Thus if $L \cap R$ has only a few elements then it is a good criterion for saying that S has a small number of keys.
In the case $L \cap R = \emptyset$, $\Omega \setminus R$ is the unique key for $S = \langle \Omega, F \rangle$ as pointed out in [1, Corollary 4].

Example We take up the example in [2, Appendix 1]

$$\Omega = \{a,b,c,d,e,f,g,h\}$$

$$F = \{a \to b,\ c \to d,\ e \to f,\ g \to h\}$$

It is clear that for this relation scheme

$$L \cap R = \emptyset,$$

and it has exactly one key, namely aceg. Taking into account the remarks 1, 2, 3 just mentioned, the above algorithm can be modified as follows.

Algorithm OL2. Set of all keys for $S = \langle \Omega, F \rangle$;

$K_S \leftarrow \{\text{Algo1}^{*)}\ (\Omega, F, (\Omega \setminus R)(L \cap R))\}$

for each K in K_S *do*

  for each FD $(L_i \to R_i)$ in $\bar{F}$ such that $K \setminus R_i \neq K$ *do*

    $T \leftarrow (K \setminus R_i) \cup L_i$;

    test ← true;

    for each J in K_S *do*

      if T includes J *then* test ← false;

    if test *then* $K_S \leftarrow K_S \cup \{\text{Algo2}^{*)}(\Omega, F, T)\}$

  end

end;

return K_S.

Remark 5 The Algorithm OL2 now has time complexity

$$O(|K_S|\,|\Omega|\,(|K_S|\,|\bar{F}| + |F|\,|L \cap R|))$$

---

*) Algo 1 and Algo 2 refer to Algorithm 1 and Algorithm 2 in [1] respectively.

R E F E R E N C E S

[1] Ho Thuan and Le van Bao, Some results about keys of relational schemas, Acta Cybernetica, Tom. 7, Fasc. 1, Szeged, 1985, pp. 99-113.

[2] Lucchesi, C.L., Osborn, S.L., Candidate keys for relations, J. of Computer and System Sciences, 17. 1978, pp. 270-279.

[3] Osborn, S.L., Normal forms for relational databases. Ph.D. Dissertation, University of Waterloo, 1977.

Néhány megjegyzés a Lucchesi-Osborn algoritmushoz

Ho Thuan

Összefoglaló

Az [1]-ben két algoritmust javasoltunk az adott szuper-kulcsban levő kulcsok megkeresésére. Ebben a cikkben, felhasználva a két algoritmust és néhány egyszerü megjegyzést, megvitatjuk a Lucchesi-Osborn féle algoritmust [2].

Несколько замечаний об алгоритме Луччеси и Осборна

Хо Тхуан

Р е з ю м е

В статье [1] мы дали два алгоритма для нахождения ключа реляционной схемы, который содержится в данном супер-ключе. В этой статье, используя эти алгоритмы и несколько простых замечаний, улучшается алгоритм Луччеси и Осборна [2].

MTA SZTAKI Közlemények 35/1986 pp.53-63

AN APPROACH TO AUTOMATED SYNTAX ANALYSIS

Lic. Miguel Fonfria ATAN
Lic. Eugenia G. Muniz LODOS
Lic. Luis A. Fajardo Alvarez de la CAMPA
Ing. Jorge L. de la Cantera RUIZ
Lic. R. Basilio Zubillaga BERAZAIN
Lic. Luis E. Fernandez LARA

Institute Central de Investigacion Digital
Habana, Cuba

Introduction.

In programming or writing compilers three problems must be solved: syntax analysis, lexicological analysis, and code generation or semantic management.

At the present level of development of the theory of languages and compilers, it is almost an established principle that from the three problems mentioned above only the third one requires a heavy treatment by the writers of compilers, i.e. the lexicon, and especially the syntax, should be a frame only for semantic management. The writer of compilers should find a syntactical method flexible enough to enable adequate semantic management.

In order to achieve the automation of syntax analyzers, a wide variety of syntax analyser generators have been developed from the description of a language grammar described in Backus's Normal Form, or in some other similar manner to provide the compiler writer with the syntax analyzer.

The method of syntax analysis can be divided into two large groups: the ascending ones which build the syntax tree of recognition from the chain been analyzed up to the generation's syntax auxiliary, and the descending ones which, starting from the generation's syntax auxiliary, work down to the text to be recognized.

Within the recognisers of the ascending type, techniques of precedence have been successful, and also that defined by Knuth in 1965, the LR technique, which has been widely accepted for theoretical and practical studies. From this, the techniques SLR and LALR, defined by De Remer in the late sixties, have been derived.

With particular reference to LALR(1) techniques, a syntax analyzer generator was implemented in 1971 starting from LALR(1) grammars, that is, grammars to which the LALR(1) technique is applicable, thus demonstrating the feasibility of generating analyzers of this type.

In 1972, a Syntax Analyzer Generator (SAG) similar to the former was built at the Centro de Investigacion Digital, from which tables were generated which enabled syntax analysis of ALGOL 60 and COBOL compilers for the CID 201-B minicomputer, the effectiveness of this method of analysis having been demonstrated in both.

The method of analysis LALR(1) in the form of a program guided by tables has the required flexibility as mentioned above. It possesses qualities which render it effective as a base for a SAG, among which are the following:

- Determinism in the analysis.
- Easy interaction with lexicologic and semantic management.
- Instantaneous detection of errors.
- Generality, in the sense that the class of LALR(1) grammars is wide.
- Easy alteration of language syntax.
- Easy recovery of syntax errors.
- Valuable parameters with respect to memory required and speed of analysis.

The method of analysis with LALR(1) grammars should be regarded as the programming of a deterministic pushdown automata that performs the syntax analysis by making adequated changes in its states.

The types of states present in this automata are three: applications, read, and look-ahead states. For each rule or production in the grammar there is a state of application, these states being the suitable ones to interact with the semantic management of the compilation process. The action that takes place in these may be resumed, from a properly syntactical viewpoint, into two tasks:

- Reducing or increasing the analyzer stack according to the number of symbols present on the right portion of the rule which produced it.
- Comparing the top of the stack with the first component of a set of associated pairs and comming up to the state indicated by the second component of the corresponding pair.

The reading states' function is to read the chain of which analysis is desired as to whether or not it belongs to the

language. The syntactical action associated with them is one of reading the symbol and comparing it with one or more that can occur in the state, comming up to the corresponding destination of the matched symbol. If the symbol being read is not among those legal in that state, a syntax error is detected.

The look-ahead states have their origin in the automata's need to look at a symbol in the chain (that is, the head) in order to determine which one is to be the next state that will enable proper continuation of the analysis. The associated syntactical action is to ascertain in what branch the looked symbol is located and come up to the corresponding destination. If the symbol is not found in any of the branches then a syntax error is detected.

These two states are the ones which interact with the lexicologic analyser in the compilation process. In conceiving the look-ahead states, it should always be remembered that these states do not read but only look at the symbol.

The method of analysis operates with a stack into which the reading states are pushed as they are consulted, the necessary history being maintained in the stack that enables continuation of syntax analysis.

SAG applications.

Our experience with the use of the SAG of CID 201-B began with the COBOL compiler written for the CID 201-B, with which the grammar corresponding to PROCEDURE DIVISION was processed. At that time the tables resulting from SAG was a listing that had to be manually loaded and optimized. Notwithstanding this

difficulty, the use of this technology increased the efficiency of the System and the speed of its setting. We will not go further into this application because the current version of SAG is an improvement by which, besides the listing with all the states, terminals, non terminals, etc., a paper tape is obtained containing the tables already coded and optimized.

In the case of the COBOL compiler for the CID 300/10 we decided to use this method for the syntax analysis of the whole language. Starting from the syntax of the COBOL sentence, the grammar for the language was designed. However, due to the structure of the compiler, which owing to memory capacity problems is divided into overlay regions, the language's grammar was divided into three parts: one for compiling IDENTIFICATION DIVISION and ENVIRONMENT DIVISION, one for compiling DATA DIVISION, and another one for PROCEDURE DIVISION. With each of these parts the SAG was used to obtain the tables and thus the corresponding syntax analyzer.

The procedure followed is simple: the compiler has an initial state to detected the symbol IDENTIFICATION, then it delivers control to the syntax analyzer of the first two divisions, which finishes upon detecting the terminal symbol DATA. Next the syntax analyzer of DATA DIVISION is loaded and given control, which ends when symbol PROCEDURE is detected, thus loading and delivering control to the syntax analyzer of this division, which finishes when end of compilation is detected.

The grammar of PROCEDURE DIVISION has a special characteristic in so far as it was necessary to divide the analyzer into two parts owing to memory capacity problems with CID 201-B when this

grammar was processed (it has the highest number of rules and is the more complex due to recursiveness). So there exists two grammars named "PROCEDURE P" and "PROCEDURE S", with which the syntax analysis of COBOL instructions is performed.

There were grouped in the "PROCEDURE P" grammar, the conditional sentences of COBOL and all grammatic rules presenting recursiveness while in the "PROCEDURE S" grammar the imperative sentences of the language were placed.

The main problem with this approach was the way in which the change from one table to another was to be carried out (in the same memory area) so that it should be transparent to the compiler. The following is a description of the solution adopted.

Syntax analyzer "PROCEDURE P" was the first to receive control, as has been explained, upon detection of the word PROCEDURE. For this analyzer, imperative sentences of the language such as ACCEPT, DISPLAY, etc., are terminals. This enables one state alone of "multiple reading" generated by the analyzer to detect the type of instruction being managed. If it was a conditional instruction, its analysis can be done within the analyzer "PROCEDURE P".

In case of an imperative instruction, when the word identifying the instruction is detected (it appears as a terminal), control is given to a special procedure (it appears in the compiler as one more semantic subroutine) that substitutes in the compiler the parameters of the tables generated by "PROCEDURE P" grammar by those of "PROCEDURE S", adequately placing certain indicators within the compiler, and continuing the processing by analyser

"PROCEDURE S". When the end of the instruction is detected. control is given to another special procedure (it appears in the compiler as a semantic subroutine) that carries out the process in reverse order, and processing then continues by analyser "PROCEDURE P".

It is to be noted that the side effect of this solution is a reduction in the speed of compilation, since the grammar tables have to be exchanged when an imperative instruction appears in the source program.

In this manner we were able to apply the SAG in the COBOL compiler for CID 300/10 and solve the problems.

With the experience obtained in applying the SAG extensive. complex grammars, we tackled the design of the compiler of the dBASE-300 laguage, having in mind to process its grammar in automated form. A grammar was designed that included all sentences in the language. This first grammar, however, could not entirely be processed by SAG. At this point we wish to comment the deficiencies and shortcomings of SAG so that the final decision adopted be understood.

The Syntax Analyser Generator (SAG) written for the minicomputer CID 201-B with 32 K words of central memory (like PDP-8), behaves like an independent program, that is, it operates with no Operating System. Due to memory restrictions, it was not possible to go deep into the diagnosis of errors; both the syntax errors of grammar (due to ambiguity of common errors in punching) and the error of exceeded capacity causes the program to stop execution without issuing any specific error message, so the user must go into an analysis of its whole grammar and find with

a pragmatic approach of "trial and error" a solution of the problem. This is a slow, tedious process in which the grammar must be rewritten again and again, and executed by SAG.

This process took, in general, more time than foreseen and we had to make a decision to obtain our syntax analyzer through the automated method (produced by SAG) and the ad hoc method.

To achieve this end we gradually reduced the initial grammar and finally obtained a grammar that can successfully be executed by SAG, the generated tables are loaded and the rest of the sentences are processed by the ad hoc method. To accomplish this it was necessary to alter both the scanner and the syntax analyzer so that it contemplated the work with the two methods.

Every command of the dBASE-300 language is processed by SAG except those that are simple (formed by terminals only) and the arithmetic expressions which upon the syntax analyzer detecting terminals with which an arithmetic expression can be started gives control to a module called arithmetic scanner that processes the expression by ad hoc method. When detecting a symbol not belonging to the expression the arithmetic scanner returns control to the syntax analyzer delivering the read symbol as not read so as not to affect the syntactic analysis.

Conclusions.

The use of a Syntax Analyser Generator allows to increase the performance of the programming techniques in any process that requires a complex syntactic analysis.

Using SAG for CID 201-B it was possible to verify the theoretical advantages that have been described in the introduction of this paper, although due to peculiarities of the implementation these advantages are reduced. In any case, it has been shown that its application may with satisfactory results be accepted as a "partial" solution for the syntax analysis of complex grammars.

Bibliography.

1.- Aho, A.V., J. Ullman The theory of Parsing, Translation and Compiling. Volume 1: Parsing. Prentice Hall, 1972.

2.- Anderson, T., J. Eve, J.J Horning. Efficient LR(1) parsers. University of Toronto, Computer Systems Group, 1972.

3.- De Remer, F.L. Simple LR(k) Grammars. Comm. ACM 14, 1971.

4.- Fonfria, M., E. Muniz, Informe tecnico del Compilador de COBOL para la CID 201-B.

5.- Fonfria, M. y otros, Cinco anos de Aseguramiento de Programas Basico de Gestion para el Sistema CID-1310, ICID, 1986.

6.- Fonfria, M. y otros, Sistema de Gestion de Bases de Datos dBASE-300. ICID, 1986.

7.- Gries, D. Compiler Construction for Digital Computers, New York. John Wley, 1971.

8.- Jares, L.R. A Syntax Directed Error Recovery Method. University of Toronto, Computer Sysytems Group, 1972.

9.- Lalonde, W.R. An Efficient LARL Parser Generator, University of Toronto, Computer Systems Group, 1971.

10.- Mc Keeman, W.M., J.J Horning, D.B. Wortma. A Compiler Generator. Englewood Cliffs, N.J., 1970.

11.- Muniz, E., V. Llopis, B. Zubillaga. Informe Tecnico del Generador de Analizadores Sintactico LALR(1), CID, 1976.

12.- Zubillaga, B. Generador de tablas optimizadas LALR(1), CID, Universidad de la Habana 1976 Tesis de Especialista.

13.- Zubillaga, B. Representacion optimizada de una Familia de Conjuntos. Rev. CID Electronica y Proceso de Datos en Cuba. No. 2 1982.

Az automatikus szintax-analizis egy tárgyalása

M.F. Atan, E.G. Lodos, L.A.F.A. de la Campa,
J.L. de la Cantera Ruiz, R.B.Z. Berazain,
L.E.F. Lara

Összefoglaló

1972-ben a Havannai Számitástechnikai Intézetben /Centro de Investigacion Digital/ épitettek egy szintax-analizis generátort /Syntax Analyzer Generator, SAG/, amely felhasználásával végezték el a CID 201-B mini-számitógép számára az ALGOL ill. COBOL forditóprogramjainak szintaktikus elemzését. A cikk a munka elméleti hátterét és a tapasztalatokat foglalja össze.

Подход к автоматическому синтакс-анализатору

М.Ф. Атан, Е.Г.М. Лодос, Л.А.Ф.А. де ла Кампа,
Й.Л. де ла Кантера Руиз, Р.Б. Беразаин,
Л.Е.Ф. Лара

Р е з ю м е

В 1972 г. был в Гаванском Вычислительном Институте /Centro de Investigacion Digital/ построен генератор синтакс-анализа /Syntax Analyzer Generator, SAG/, с помощью которого были анализированы компайлеры АЛГОЛ-а и КОБОЛ-а на машине CID 201-B. Статья описывает теоретические основы работы и опыты с работой.

MTA SZTAKI Közlemények 35/1986 pp. 65-77

THE AUTOMATIC GENERATION SYSTEM OF PATTERN RECOGNITION PROCEDURES RECLASS-COTO

HO TU BAO - HOANG KIEM

Institute of Informatics and Cybernetics
Hanoi, Vietnam

1. INTRODUCTION

During recent years, research in the field of artificial intelligence has produced effective new techniques for representing empirical judgement knowledge and using this knowledge in performing plausible reasoning These techniques are concerned with how we should go about designing problem solving systems. Expert systems are problem-solving programs that solve substantial problems requiring expertise. An expert system is usually made of two parts:

- a knowledge base
- an inference engine

The knowledge base can be a set of rules, which represent part of the knowledge of experts in a given domain, and a set of facts described, a particular situation to be analysed.

The inference engine is the rule interpreter, i.e. the program which understands the rules and uses them to produce new facts. It is rather independent of the domain of the knowledge base, and depends on the syntax of knowledge representation (5-6).

A major evolution that has occured during the last years is to view the design of a pattern recognition as

a highly iterative process in which pattern analysis is considered as an intrinsic and important part of the design process.

Based on the model of the general pattern recognition problem (1-4) and the techniques of expert systems, we have designed an automatic generation system of pattern recognition procedures RECLASS-COTO. This system is used for solving the various problems in Vietnam.

2. DESCRIPTION OF THE SYSTEM R E C L A S S

In the first design, the automatic generation system of pattern recognition and classification procedure consists of the following subsystems:

SDATA: Input starting structured data and the orders issued from the users (prior information defining the type of problem), criterion for evaluating results (final structured data, transformations of data, recognition and classification algorithms...)

ISDA: Interpretations on the basis of the starting structured data for choosing the suitable subprocedures (it is necessary in the case the orders issued from the users are missing.)

We can build various SDATA and ISDA for the package RECLASS depending on the type of starting structured data, for example, the starting structured data represented by a vector X, or more precisely by a point in R^n, by a set of lists, graphs,... First we choose the type of input based on the model of the general pattern recognition problems (15). Correspondingly, there are SDATA1 and ISDA1.

TRAS: Transformations of the starting structured data such as:

- Linear and nonlinear transformations to map patterns to lower-dimensional spaces for pattern representation or to enhance separability between classes.
- Feature evaluation criteria.
- Subprocedures for suboptimal selection of a subset from a given set of a measured or derived features.

We have been building some heuristic subroutines as follows:

TRAS1: Fourier, Fast Fourier Transforms

TRAS2: Karhunen-Loève Transforms

TRAS3: Walsh-Hadamard Transforms

TRAS4: Haar Transforms

TRAS5: Bhattacharyya Transforms

TRAS6: Kruskal Transforms

...

CLAR: The basic Recognition and Classification algorithms

CLAR1: NNR algorithms (Nearest Neighbour Rule)

CLAR2: PF algorithms (Potential Function)

CLAR3: CV algorithms (Calculation of Valuation)

CLAR4: DF algorithms (Distribution Function)

CLAR5: PP algorithms (Projection on the Plan)

CLAR6: LH algorithms (Linear Hyperplane)

CLAR7: E algorithms (Entropy algorithms)

CLAR8: K-MEANS algorithm

CLAR9: ISODATA algorithm

CLAR10: MND algorithms (Methods des Nuees Dynamiques)

...

MODIA: Modifications (transformations) of algorithms

Based on results presented in the above sections, we have been building the following subroutines:

MODIAN: Normalization
MODIAC: Combination
MODIAP: Parametrization
MODIAM: Multicriteria
MODIAF: Fuzzy Modifications

...

METRIC: The package RECLASS uses the following metrics:
METRI1: Minkowsky metric
METRI2: Hamming metric
METRI3: Camberra metric
METRI4: Quadratic metric
METRI5: Chebyshev metric
METRI6: Mehalanobis metric
METRI7: "city block" metric
METRI8: 2 metric (Benzecri)
METRI9: Seller metric

...

EVA: Interpretation on the basis of the final structured data. Some heuristic evaluation functions may be chosen in the following:

EVA1: Minimization of the mean risk (Bayes procedures)
EVA2: Maximization of the variance between classes
EVA3: Minimization of the variance within classes
EVA4: Stabilization of neighbourhood relation
EVA5: Stabilization of results (f.s.d) by leave-one-out method

...

F DATA: Output final structured data

It is clear that, ISDA play an important role for automatic generation of the pattern recognition and classification procedures. In practice, users can usually provide a training or learning set, i.e an equivalent (or approximity, fuzzy...) relation as well as the knowledge base for choosing the suitable subprocedures. Then, ISDA propose the choice based on the following inference engine.

3. THE INFERENCE ENGINE COTO

COTO is an domain-independent inference engine had been designed by us. In COTO, knowledge is represented in the form of the production rules:

```
RULE i
    IF   premise 1
    and  premise 2
    ..............
    and  premise n
    THEN conclusion 1
    and  conclusion 2
    ..............
    and conclusion  m
```

The keywords used by the system are:

- 'RULE' following by the number of rule
- 'IF' beginning of premises in the rule
- 'AND' join operator of premises and conclusions
- 'THEN' beginning of conclusions in the rule
- 'NON' negation operator of a proposition

and the comparison and affectation operators of a numerical variable:

- '>' greater than
- '> =' greater or equal
- '>' less than
- ' =' less or equal

- '<>' different
- '↔' affectation of a value to a numerical variable and the comparison and affectation operators of an alphanumerical variable:

- '==' equal
- '><' different
- ':=' affectation of a value to an alphanumerical variable

A <premise> in general form corresponds to a triplet:

<object 1><Relation><object 2>

where <relation> corresponds to a comparison operator of a numerical or alphanumerical variable.

<object 1> can take one of the following forms:

- A proposition, i.e. sequence of words without keywords, except "not" standing for negation. It may be taken a logic value true or false.

<object 1> is considered as a proposition if and only if there are not two terms <Relation> and <object 2>

- A numerical and alphanumerical variable substituable by one of initial facts or reduced facts in answering the question in conversation.

The nature of variables (numerical or alphanumerical) is distinguished by the comparison operators. The numerical variables can take any real value. Each alphanumerical variable can only take the values in the set of attributes determined in the lecture of the rule set. These attributes are considered exclusive.

- A name of a numerical function at its values can be evaluated in engining. This name is determined as the head of a subroutine i.e. a name and a list of variables between two parentheses.

In practice, a function is considered as a numerical variable but its value may be evaluated only by an extern procedure.

<object 2> can take one of the following forms:

- A numerical or alphanumerical constant as the second operand when exists <relation>
- A numerical variable in the case of comparison of two numerical variables.
- The name of a numerical function. This case can appeared only when the <object 1> is a function.

A <conclusion> may be represented in the form:

<affected><op. affectation><affectation>

where <op. affectation> correspond to one of the following operators

'←=' affectation of a value to a numerical variable

':=' affectation of a value to an alphanumerical variable.

<affected> may be one of the followings

- a proposition in the case there are not two last terms

  <op. affectation> and <affectation.>

- a numerical or alphanumerical variable
- a name of a numerical function
- a cell to a subroutine

  <affectation> corresponds to a numerical or alphanumerical constant or a function which may be evaluated.

4. THE STRATEGIES OF REASONING

There are two basic strategies in COTO: Forward and Backward reasonning.

Suppose that F and R are the base of facts and the base of production rules. There are several classical ways of reasoning: the two most current ones are called MODUS PONEN and MODUS TOLLENS.

The MODUS PONENS is defined as follows:

if $(X \supset Y) \in R$

and $X \in F$

then $Y \in F$

This is usual way to produce new facts using the rules. There is another one, based on a well known property in logic, the MODUS TOLLENS:

if $(X \supset Y) \in R$

and $\neg Y \in F$

then $\neg X \in F$

Two principal operations are often utilised

- Activation of a rule by the modus ponens and modus tollens
- Disactivation of a rule:

a) By modus tollens

if $(X \supset Y) \in R$

$\neg Y \in F$

then $R := R \setminus \{(X \supset Y)\}$

b) if $(X \supset Y) \in R$

$\neg X \in F$

then $R := R \setminus \{(X \supset Y)\}$

c) if $(X \supset Y) \in R$

$Y \in F$

then $R := R \setminus \{(X \supset Y)\}$

As in many Inference Engines, two basic strategies of reasonning are used in COTO:

- Forward reasonning

One try to deduce all of facts which may be deduced. This consist in looking for rules which have premises known to be true (i.e. which are facts) and to activate them (i.e. to use them to produce new facts). This is performed until no more are activable.

- Backward reasonning

It is the reverse operation. Let us suppose we want to prove that the statement 6 is true: then 6 becomes a GOAL.

If we consider the rules which result in a goal, we can

add their premises to the tree of goals (as SUBGOAL). When a subgoal is not the conclusion of a rule, it must be a fact, else the final goal is not provable.

Backward reasonning can be interesting in two ways:

- to prevent from producing too much facts which may be useless regarding the goal requested by the user.
- to ask questions to the user in order to complete the set of facts until the goal is proved.

5. SOME REMARKS AND APPLICATIONS

- The system RECLASS-COTO may run under the operational systems for the computers ODRA-1304, IBM 360/40, IBM 360/50, MICRAL, IBM-PC. The users need only to provide the starting structured data and the answers for the conventional questions, for example:

MODE "= P (a prior choice of procedures)

PROBLEM = R(N) (recognition problem of type N)

CRITERION = MIN(RISK) (recognition problem of type N)

METRIC = = (Euclidean metric to be used)

ALGORITHM = C(MNR, PF, CV, DF) i.e. combination of the algorithms NNR, PF, CV, DF

...

In the case when MODE = A, i.e. an automatic generation of procedures occurs, subprocedures are built by guiding of inference engine COTO based on the knowledge base for choosing the efficient subprocedures. This knowledge base consists of 115 rules, for example:

```
... RULE 61

    IF   METHOD == CLASSIFICATION
         AND FIGURES = 1
         AND CLASSER == THE VARIABLES
         AND CLASSIFICATION TYPE == HIERACHIE
         THEN ORDER := CRSDVA
         AND ORDER := WARD
         AND ORDER := SAUT
         AND METRIC = E.

... RULE 78

    IF   FIGURE = 4
         AND METHOD == FACTORIELLE
         THEN ORDER := SISPAD
         AND interface with SPAD or MODULAD
         AND MULTI-CORRESPONDENCES ARE PROPOSED.
```

In the last year, we have used the RECLASS-COTO for solving the following problems:

- Recognition of the oil-producing structures.
- Interpretation and classifications of the data of geochemistry, gravity, magnetic and seimic surveys for geological and geophysical maps.
- Classifications of the ore deposits and selection of the significant features for various categories of the ore deposits.
- Recognition and classifications of the information from numerous aerial photographs and satellite data.

The results obtained indicated other promissing applications of the system RECLASS-COTO. We hope that further extention for more generality and more realistic learning abilities of the system can be achieved by combining the recognition algorithms and the techniques of expert system.

REFERENCES

1. KANAL, L.: Patterns in Pattern Recognition, 1968-1974. IEEE Trans. on Information Theory. Nov. 1974, pp. 697-722.

2. SIMON, J.C.: Recent progress to a formal of pattern recognition and scene analysis. Rapport de recherche N°.76, IRIA, 1974.

3. TOU, I.T. - GONZALEZ, R.C.: Pattern recognition principle. 1974.

4. Журавлёв Ю.И. и другие: Задачи распознавания и классификации со стандартной обучающей информацией. ЖВМ № 5,1980, pp.1294--1309.

5. LAURIERE, J.L.: Les systems experts. Technique et Science Informatique N°.1, 1982.

6. PINSON, S.: Representation des connaissances dans les systems experts. RAIRO, N°.4, 1981, pp. 343-367.

RECLASS-COTO: egy alakfelismerési eljárásokat automatikusan generáló rendszer

Ho Tu Bao - Hoang Kiem

Összefoglaló

Szakértő rendszerek olyan feladat-megoldó programcsomagok, amelyek nehéz tapasztalati tényeket is igénylő problémákat oldanak meg. Egy szakértő rendszer általában két részből áll: tudás-bázisból és az u.n. "inferencia-gépből" /inference engine/.
Az utóbbi tulajdonképpen egy szabály-fordító /interpretáló/ program, azaz olyan program, amely megérti a szabályokat és felhasználja azokat uj tények alkotásához.
A RECLASS-COTO rendszert, amely egy általános alakfelismerési eljáráson, valamint szakértői rendszerek technikáján alapszik, Hanoiban fejlesztették ki és számos probléma megoldásában már sikeresen alkalmazták.

RECLASS-COTO: система автоматического генерирования процедур для решения проблем распознания образов

Хо Ту Бао, Хоанг Кием

Р е з ю м е

Экспертные системы, это пакеты программ, которые предназначены для решений тяжелых проблем, где нужны тоже опытные факты. Экспертная система состоит из двух частей: базис знания /knowledge base/ и "инференце-машина" /inference engine/. Инференце-машина есть в сущности программа для интерпретации правил, которая понимает правила и использует их для вывода новых фактов. Система RECLASS-COTO основана на общей процедуре распознания образов и на технике экспертных систем, была разработана в Ханойе и применили ее к решению многих практических проблем.

MTA SZTAKI Közlemények 35/1986 p. 79-87

INDEX TREATMENT OF A DBMS FOR A MINICOMPUTER

Lic. Maria Elena Bragado BRETANA
Lic. Miguel Fonfria ATAN

Institute Central de Investigacion Digital
Habana, Cuba

Introduction

In this paper it is shown the method implemented for index treatment in the Data Base Management System dBASE-300, for the cuban minicomputer CID-300/10 (like the PDP-11/05).

The technique employed to treat indexes in dBASE-300 is the organisation called B+ tree. With the use of this technique it is possible to accomplish all transactions defined in dBASE-300 in a suitable way as it is efficient for both random and sequential access to records and modification operations.

In order to have a better understanding of the implementation of B+ tree in dBASE-300 File Control System, the analysis is divided into the following sections:

- B-trees and their data structures.
- Find, Insertion and Deletion.
- Operation costs.
- B+ trees.
- Other variants of B-trees.

The first two sections are referred to B-trees because the characteristics explained are also present in B+ trees.

B-trees and their data structures.

The structure implemented for index treatment in dBASE-300 corresponds to the B-tree definition given in [Bay], i.e., it has the following properties:

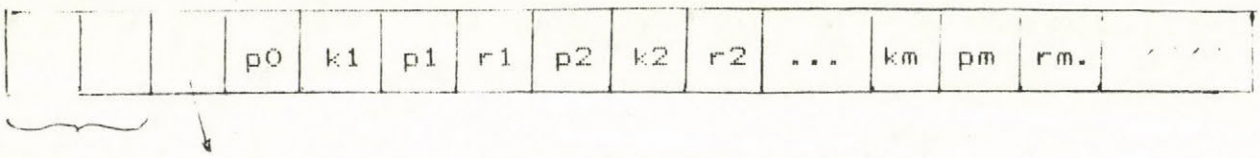
a) Each path from the root to any leaf has the same length.

b) Each node, except the root and leaves, has at least k+1 sons, where k is a natural number. The root is a leaf and has at at least two sons.

c) Each node has at most 2k+1 sons.

A node of a B-tree is the page in which the index is stored.

A page of the B-tree in dBASE-300 looks like this:



next & previous page

number of keys in the node

pi - pointer
ki - key
ri - record number

This data structure of a page has the following properties [Bay]:

a) Each page contains between k and 2k keys except the root page which may contain between 1 and 2k keys.

b) Let m be the number of keys on a page (not a leaf). P has m+1 sons.

c) The keys in a page are sequential in increasing order and each page contains at most 2k keys and 2k+1 pointers.

d) Let P(pi) be the page to which pi points, let K(pi) be the set of keys on the pages of that maximal subtree of which P(pi) is the root. Then the following conditions always hold:

$\forall\ y \in K(p0) \Rightarrow y<k1$

$\forall\ y \in K(pi) \Rightarrow ki<y<ki+1 \quad ; \quad 1 \leq i<m$

$\forall\ y \in K(pm) \Rightarrow km<y$

The record number behind each key in the page enables to access directly the requested record in the relative file.

This information in the page is a peculiar characteristic of this implementation, taking advantage of the relative organisation of the data file.

Find, Insertion and Deletion

Find.

A find operation in a B-tree of order k never visits more than $1+\log_k n$ pages, where n is the number of records in the file.

This is possible because record operations in a B-tree always leave the tree balanced.

The B-tree balancing scheme restricts changes in the tree to a single path from a leaf to the root, so it can not introduce "runaway" overhead [Com].

The find algorithm is simple logically.

Insertion

The insertion operation requires a previous find operation with which it is possible to get the page where the new key is going to be inserted.

Three cases may be found (assuming the new key is not present):

i) empty tree

ii) page not full

iii) page full

In the first case it is necessary to create a root page with the new key.
In the second case the new key may be inserted in the correct position.

In the third case it is necessary to split the page, that is, the smallest k keys are placed in one page, the largest k keys are placed in another page, and the remaining value is promoted to the parent page where it serves as a separator. In the worst case splitting propagates all the way to the root and the tree increases in height by one level.

Deletion

Like insertion, the deletion operation needs a previous find operation to locate the key to be deleted.
Assuming the key is present, it is possible to find two cases:

a) the key is on leaf page

b) the key is not on a leaf page.

In the first case the key can be deleted from leaf.
In the second case it is necessary to find the adjacent key and place it in the position of the deleted key. To find the adjacent key in key-sequence order it is necessary to search for the leftmost leaf in the right subtree of the deleted key.

In both cases it is necessary to check whether an underflow condition is present, that is, if the leaf has less than k keys. In this case it is possible to redistribute the remaining keys between the two neighboring pages but only if there are at least 2k keys to distribute. If there are less than 2k keys it is necessary to perform a concatenation process, where keys are combined into one page and the other is discarded. Then the separating key in the ancestor is no longer necessary and is also added to the single remaining leaf.

As can be seen, the process of concatenation may force concatenating at the next higher level and so on, to the root of the level, and it is possible that B-tree decreases in height by 1.

Operation costs

The cost of a find operation is increased with the growth of the file size logarithm. This is:

$$h \leq \log_k \frac{n+1}{2} \qquad \text{[Bay] [Com]}$$

where : h - height of the page tree

n - number of keys

k - minimum number of keys per page

Insertion and deletion take time proportional to $\log_k n$ in the worst case because these operations need aditional access beyond the cost of a find operation as it progresses back up the tree. The costs are at most double, so that the height of the tree still is the main element for these costs.

As can be seen, the number of keys in a page is the parameter on which the performance of all operations depends.
Bayer and McCreight [Bay] show a way to determine the optimal number of keys in a page to achieve minimum time per transaction, in terms of fixed time spent per page, transfer time, key size, and a factor for average page accupancy.

In the case of dBASE-300 there are practical limits to page size. The disks on which the system is supported are divided into fixed blocks of 512 bytes length.
Thus, each page in dBASE-300 is 512 bytes in order to avoid extra overhead in the process.
In practice, considering standard key size the behaviour of the algorithm with this size page was quite suitable.

B+ trees

The organisation chosen was B+ tree, a variant that avoids the problems with sequential processing in B-trees.
These problems are the requirement of extra space for at least $\log_k (n+1)$ pages [Knu] in main memory to avoid reading them twice in the sequential processing.
Also, the next operation may require to access several pages before finding the desired key.
In a B+ tree all keys reside in the leaves.
The relative organisation of a file in the dBASE-300 enables implementing B+ trees in a very comfortable way, because this file can be accessed directly once we have the number of correspondent record in the sequence order.

With B+ tree the logarithmic cost properties for operations by key are retained and the next operation for a sequential processing requires at most 1 access. Besides, no page will be accessed more than once.

On the other hand, the feature of B+ trees that all keys reside in the leaves allows to simplify the deletion operation because the removal of the key to be deleted is simple, and the tree need not be changed while the leaf remains al least half-full. It means that a copy of a deleted key will be present in the tree and can direct searches to the correct leaf.

Redistribution or concatenation process will be necessary only if an underflow condition arises in the leaf and this process is similar to that in B-tree.

Other variants of B-trees

Several variants of B-trees were analyzed in order to choose a technique for the implementation of index treatment in dBASE-300. These variants were: B* tree [Knu], Virtual B-trees [Com], Compression technique [Wagn], and Binary B-trees [Bay1].

B* tree was discarded because its implementation seems to be a little more complicated and then it requires more main memory with the resulting increase of overhead.

In the case of Virtual B-trees there is a practical inconvenience because our system computer does not have facilities of virtual memory.

With the compression technique pointers can be compressed using a displacement from a page address instead of the absolute address

value. This technique is particulary useful for virtual B-trees where pointers take on large address values.

Finally, Binary B-trees are appropriated for a one-level store, bacause a Binary B-tree is a B-tree of order 1. This means that each page has 1 or 2 keys and 2 or 3 pointers.

Conclusions

B+ trees retains the logarithmic cost properties for operations by key and has no problems with the next operation for a sequential processing.

In practice, B+ tree proved a very suitable organisation for index ,treatment in dBASE-300.

References

[Bay] Bayer, R. and Mc Creight. "Organization and Maintenance of Large Ordered Indexes". Acta Informatica 1,3(1972) 173-189.

[Bay1] Bayer, R. "Binary B-trees for Virtual Memory". Proc. 1971 ACM SIGFIDET Workshop, ACM New York, (219-235).

[Bel] Bell, D.A. and S.M. Deen. "Hash trees versus B-trees". The Computer Journal Vol.27, No. 3 1984.

[Bla] Black, J.P. et al. "A robust B-tree implementation". Proceedings of the Fifth International Conference on Software Engineering. pp 63-70. (9-12 March 1981).

[Com] Commer, D. "The ubiquitous B-tree". Computer Surveys, Vol. 11, No. 2, June 1979.

[Knu] Knuth, D. "The Art of Computer Programming", Vol. 3.

[Wag] Wagner, R. "Indexing designs considerations". IBM Syst. J.4 (1973) 351-367

Miniszámitógépek számára irt DBMS index-kezelése

M.E. Bragado Bretana, M.F. Atan

Összefoglaló

A cikkben a kubai CID-300/10 /∿ PDP-11/05/ mini-számitógép számára irt dBASE-300 adat-kezelő rendszer index-kezelését ismertetik. Az index-kezelési technika "B+tree"-nek nevezett szervezésen alapszik.

Трактовка индексов в DBMS для мини-компьютеров

М.Е. Брагадо Бретана, М.Ф. Атан

Р е з ю м е

В статье речь идет о трактовке индексов в имплементации пакета обработки данных dBSE-300 для кубинского мини-компьютера CID-300/10 /~ PDP-11/05/. Техника трактовки индексов основана на организации названной "B+tree".

MTA SZTAKI Közlemények 35/1986 p. 89-104

DUALITY IN DYNAMIC PROGRAMMING

DINH THE LUC

Computer and Automation Institute
Hungarian Academy of Sciences

1. INTRODUCTION

To give an idea about duality in mathematical programming Hadley (1962) cited the following line from "The Rubayiat" of E. Fitzgeral:

> "All of this of Pot and Potter-Tell me
> Who is the Potter, pray, and who is the Pot?"

The highlight of duality lies in the reciprocal relations lingking "Pot" and "Potter". To see it let us remember the case of linear programming. Given a linear (minimization) problem, called primal, there is a second, corresponding (maximization) problem, called dual such that

(i) The dual of the dual is the primal

(ii) Any feasible solution of the primal has a value greater or at least equal to the value of any feasible solution of the dual.

(iii) If an optimal solution exists for the primal then one exists for the dual and their optimal values are equal.

(iv) If the primal has an unbounded solution, the dual has no feasible solution and vice versa.

These features have proved to be very important in the theoretical study of optimization problems and their computational practice. For the case of nonlinear programming similar

results can be obtained, but of course, suitable conditions have to be imposed. What about case of dynamic programming? To our knowledge very little attention has been directed to this topic (some efforts towards it have been made by using Lagrangian multipliers in solving dynamic problems, (see Bellman (1957)) and a duality theory has been constructed for infinite horizon optimization of linear and convex models (see Weitzman (1973), Evers (1983) and Haneveld (1985)).

It is not the aim of the paper to develop the complete duality theory for dynamic programming. Our purpose is merely to study some duality aspects of a finite-stage dynamic system and in doing so we call attention of researchers to a useful technique for solving dynamic problems. The paper is structured as follows: in the second section we describe a dynamic problem to be considered. The third section is devoted to the Lagrangian functions associated with our problem and their properties. In section 4, another approach to duality theory, a conjugate function approach is presented and finally we give a brief discussion about the possibility of applying duality results in solving the problem.

2. DESCRIPTION OF THE MODEL

Suppose that we have a dynamic system which consists of N stages numbered from 1 to N. At stage $k \in \{1,\ldots,N\}$ the system is characterized by a nonempty set of states X_k and a nonempty set of actions $U(x_k)$ corresponding to every $x_k \in X_k$. In the case $k \leq N-1$, under action

$$u_k \in U(x_k) \tag{1}$$

state

$$x_k \in X_k \tag{2}$$

will be transferred into state $x_{k+1} \in X_{k+1}$ of the next stage by the law:

$$x_k = g_k(x_k, u_k). \tag{3}$$

Let us start with x_k and apply action u_k to get x_{k+1}. Keeping in mind requirements (1), (2) and (3) and carrying on the above procedure we get a sequence of states $(x_k,\ldots,x_N)$ and a sequence of actions $(u_k,\ldots,u_{N-1})$. Such sequences of states and actions are called (N-k)-process and (N-k)-policy, respectively. It is obvious that when x_k and u_k are chosen, x_{k+1} is completely defined.

Further, for any (N-k)-process and (N-k)-policy, let $R_k(x_k,\ldots,x_N,u_k,\ldots,u_{N-1})$ be the cost of getting $(x_k,\ldots,x_N)$ by using policy $(u_k,\ldots,u_{N-1})$.

The problem that we are going to deal with is to find an N-policy for a given initial state x_1 such that it minimizes

$$R_1(x_1,\ldots,x_N,u_1,\ldots,u_{N-1})$$

over all possible N-processes starting with x_1 and all possible N-policies.

For the sake of simplicity we shall assume that the additivity property holds for $R_1,\ldots,R_N$, i.e.

$$R_k(x_k,\ldots,x_N,u_k,\ldots,u_{N-1}) = \sum_{i=1}^{N-1} f_i(x_i,u_i) + f_N(x_N)$$

where $k = 1,\ldots,N-1$ and $R_N(x_N) = f_N(x_N)$.

It is well known (see for example Bellman (1957)) that under the additivity assumption Bellman's equations will be satisfied:

$$B_k(x_k) = \min\{f_k(x_k,u_k) + B_{k\,1}(g_k(x_k,u_k)) : u_k \in U(x_k)\}$$

where $k = 1,\ldots,N-1$ and $B_N(x_N) = f_N(x_N)$.

Finally, it will be assumed in the model that constraints $u_k \in U_k$ are explicitly expressed by relations:

$$h_k(x_k,u_k) \leq 0, \quad k = 1,\ldots,N-1 \tag{4}$$

where u_k is taken from an arbitrary space U. For the sake of convenience we will assume that $h_1,\ldots,h_{N-1}$ are scalar--valued, although all the results to be proved are valid for the vector-valued case too.

3. LAGRANGIAN FUNCTIONS

For every $k \in \{1,\ldots,N\}$ and $x_k \in X_k$ let us consider the following problem denoted by P(k):

$$\min \left[\sum_{i=k}^{N-1} f_i(x_i,u_i) + f_N(x_N) \right]$$

$$\text{s.t.} \quad x_{k+1} = g_i(x_i,u_i)$$

$$h_i(x_i,u_i) \leq 0, \quad i = k,\ldots,N-1.$$

Whenever k is indicated, we shall write x and u instead of $(x_{k\,1},\ldots,x_N)$ and $(u_k,\ldots,u_{N-1})$, respectively and

$$\lambda = (\lambda_k,\ldots,\lambda_{N-1}), \quad \mu = (\mu_k,\ldots,\mu_{N-1})$$

are (N-k-1)-vectors of real numbers. In what follows, we will not speak of the dimensions of the variables if it is clear from the context.

Associated with $P(k)$ the lagrangian function will be:

$$L_k(x_k,x,u,\lambda,\mu) = \sum_{i=k}^{N-1} [f_i(x_i,u_i) + \lambda_i h_i(x_i,u_i) + \mu_i(x_{i+1}-g_i(x_i,u_i))] + f_N(x_N),$$

if $\lambda \geq 0$ and

$L_k(x_k,x,u,\lambda,\mu) = -\infty$ otherwise.

Definition 3.1 For a fixed $x_k \in X_k$, $(x^*,u^*,\lambda^*,\mu^*)$ is said to be a saddlepoint for L_k if

$L_k(x_k,x,u,\lambda^*,\mu^*) \geq L_k(x_k,x^*,u^*,\lambda^*,\mu^*) \geq L_k(x_k,x^*,u^*,\lambda,\mu)$

for every x,u,λ and μ.

Lemma 3.1 $(x^*,u^*,\lambda^*,\mu^*)$ is a saddlepoint for L_k if and only if

(i) $L_k(x_k,x^*,u^*,\lambda^*,\mu^*) = \min \{L_k(x_k,x,u,\lambda^*,\mu^*): x,u\}$

(ii) $x^*_{i+1} = g_i(x^*_i,u^*_i)$, $i = k,\ldots,N$,

where $x^*_k = x_k$,

(iii) $\lambda^*_i h_i(x^*_i,u^*_i) = 0$,

$h_i(x^*_i,u^*_i) \leq 0$, $i = k,\ldots,N-1$.

Proof If $(x^*,u^*,\lambda^*,\mu^*)$ is a safflepoint of L_k, then (i) is obvious. Moreover, $L_k(x_k,x^*,\lambda^*,\mu^*) =$

$$= \max \{L_k(x_k,x^*,u^*,\lambda,\mu):\lambda,\mu\}.$$

If one of the conditions in (ii) and (iii) does not hold, then by varying λ and μ suitably, $L_k(x_k,x^*,u^*,\lambda,\mu)$ may increase infinitely, which is impossible. The inverse assertion is trivial. #

Define $p_k(x_k,x,u) = \max \{L_k(x_k,x,u,\lambda,\mu): \lambda,\mu\}$,

$$d_k(x_k, \ , \ ) = \min \{L_k(x_k,x,u,\lambda,\mu): x,u\}.$$

It is easy to see that

$$p_k(x_k,x,u) = \sum_{i=k}^{N-1} f_i(x_i,u_i) + f_N(x_N)$$

if (3) and (4) hold and

$$p_k(x_k,x,u) = +\infty \quad \text{otherwise.}$$

Consequently, the problem min $\{p_k(x_k,x,u) : x,u\}$ will be the same as P(k). We call it the k-primal problem and denote its optimal value by $B_k(x_k)$.

The k-dual problem will be

$$\max \{d_k(x_k,\lambda,\mu) : \lambda,\mu\} \qquad D(k)$$

and itsoptimal value will be denoted by $C_k(x_k)$.

Proposition 3.1 (Weak duality) For every (x,u,λ,μ) we have $d_k(x_k,\lambda,\mu) \leq p_k(x_k,x,u)$.

Proof. This follows immediately from the definitions. #

Proposition 3.2 (x,u,λ,μ) is a saddlepoint for L_k if and only if (x,u) solves the k-primal problem, (λ,μ) solves the k-dual problem and their optimal values are equal.

Proof. Apply Lemma 3.1 to get this proposition. #

Proposition 3.3 Functions $d_k,\ldots,d_N$ yield the following recurrence relations:

$$d_i(x_i,\lambda,\mu) = \min\{f_i(x_i,u_i) + \lambda_i h_i(x_i,u_i) + \mu_i(x_{i+1}-g_i(x_i,u_i)) + d_{i+1}(x_{i+1},\lambda',\mu'): x_{i+1},u_i\}$$

if $\lambda_i \geq 0$ and $d_i(x_i,\lambda,\mu) = -\infty$ otherwise;

$i = k,\ldots,N-1$ and $d_N(x_N) = f_N(x_N)$,

where the sign "'" denotes a vector for which the first component is omitted.

Proof. If $\lambda \not\geq 0$, then by definition $d_i(x_i,\lambda,\mu) = -\infty$ and there is nothing to prove. Now assume $\lambda \geq 0$. Consider the following dynamic model consisting of stages numbered from k to N. The sets of states are the same as in the model described in Section 2. For every state $x_i \in X_i$ $(i \geq k)$, the set of actions $U(x_i) = (X_{i+1},U)$. Action (x_{i+1},u_i) transfers state x_i into a state of the next stage by the relation:

$$G_i(x_i,(x_{i+1},u_i)) = x_{i+1}.$$

The cost of getting $(x_k,\ldots,x_N)$ by using policy $((x_{i+1},u_i),\ldots,(x_N,u_{N-1}))$ will be

$$\sum_{i=k}^{N-1} F_i(x_i,(x_{i+1},u_i)) + F_N(x_N),$$

where $F_i(x_i,(x_{i+1},u_i)) = f_i(x_i,u_i) +$

$+ \lambda_i h_i(x_i,u_i)$

$+ \mu_i(x_{i+1}-g_i(x_i,u_i))$,

$i = k,\ldots,N-1$ and $F_N(x_N) = f_N(x_N)$.

Apply Bellman's equations to this model to get the following recurrence relations:

$$B_i(x_i) = \min\{F_i(x_i,(x_{i+1},u_i)) + B_{i+1}(G_i(x_i,(x_{i+1},u_i))) : (x_{i+1},u_i)\}$$

$i = k,\ldots,N-1$ and $B_N(x_N) = F_N(x_N)$.

This gives the relations of the proposition. The proof is complete. #

Corollary 3.1 The following relation holds

$$d_k(x_k,\lambda,\mu) = \min\{f_k(x_k,u_k) + \lambda_k h_k(x_k,u_k) - \mu_k g_k(x_k,u_k)) : u_k\} + \min\{\mu_k x_{k+1} + d_{k+1}(x_{k+1},\lambda',\mu') : x_{k+1}\}.$$

Proof. This follows immediately from Proposition 3.3. #

Proposition 3.4 If $(\bar{x},\bar{u},\bar{\lambda},\bar{\mu})$ is a saddlepoint for L_k with initial point $\bar{x}_k$, then $(\bar{x}',\bar{u}',\bar{\lambda}',\bar{\mu}')$ is a saddlepoint for L_{k+1} with initial point $\bar{x}_{k+1} = g_k(\bar{x}_k,\bar{u}_k)$.

Proof. By virtue of Lemma 3.1 it suffices to show that

$$L_{k+1}(\bar{x}_{k+1},\bar{x}',\bar{u}',\bar{\lambda}',\bar{\mu}') = \min\{L_{k+1}(\bar{x}_{k+1},x',u',\bar{\lambda}',\bar{\mu}'):x',u'\}. \tag{5}$$

Indeed, again by Lemma 3.1, $\bar{\lambda}_k \geq 0$ and $\bar{x}_{k+1} = g_k(\bar{x}_k,\bar{u}_k)$.

It follows from Proposition 3.3 that

$$\begin{aligned} d_k(\bar{x}_k,\bar{\lambda},\bar{\mu}) = {} & f_k(\bar{x}_k,\bar{u}_k) + \bar{\lambda}_k h_k(\bar{x}_k,\bar{u}_k) + \\ & + \bar{\mu}_k(\bar{x}_{k+1}-g_k(\bar{x}_k,\bar{u}_k)) + \\ & + d_{k+1}(\bar{x}_{k+1},\bar{\lambda}',\bar{\mu}'). \end{aligned} \tag{6}$$

Remembering the definition of d_k we have

$$\begin{aligned} d_k(\bar{x}_k,\bar{\lambda},\bar{\mu}) &= L_k(\bar{x}_k,\bar{x},\bar{u},\bar{\lambda},\bar{\mu}) = \\ &= f_k(\bar{x}_k,\bar{u}_k) + \bar{\lambda}_k h_k(\bar{x}_k,\bar{u}_k) + \\ &\quad + \bar{}_k(\bar{x}_{k+1}-g_k(\bar{x}_k,\bar{u}_k)) + \\ &\quad + L_{k+1}(\bar{x}_{k+1},\bar{x}',\bar{u}',\bar{\lambda}',\bar{\mu}'). \end{aligned}$$

Compare the latter relation and (6) to get (5). The proof is complete. #

Corollary 3.2 If $B_k(x_k) = C_k(x_k)$, then $B_i(x_i) = C_i(x_i)$ for all $i \geq k$, where $x_{i+1} = g_i(x_i,u_i)$

u_i solves $B_i(x_i)$.

Proof. If $B_k(x_k) = C_k(x_k)$, then any optimal solutions of P(k) and D(k) form a saddlepoint for L_k. Now the assertion implied by Proposition 3.4. #

In the rest of the section, we are concerned with the question of the existence of λ and μ for a given optimal solution (x,u) of P(k) so that (x,u,λ,μ) forms a saddlepoint for L_k. As it is known from the theory of mathematical programming, the answer to this question is not always positive. For λ and μ to exist some more assumptions about the model are needed. As before, let x_k be fixed from x_k.

Proposition 3.5 Assume that the following conditions hold:

(i) There are $u_k,\ldots,u_{N-1}$ such that $h_i(x_i,u_i) < 0$ with $x_{i+1} = g_i(x_i,u_i)$, $i = k,\ldots,N-1$

(ii) $g_i(x_i,u_i) = V_i)x_i) + W_i(u_i) + c_i$,

where V_i's and W_i's are linear maps and c_i's are vectors

(iii) $f_k,\ldots,f_N$ and $h_k,\ldots,h_{N-1}$ are convex.

Then for every optimal solution (x,u) of P(k) there are λ and μ such that (x,u,λ,μ) is a safflepoint for L_k.

Proof. Consider the following problem

$$\min f(y)$$
$$\text{s.t. } My = b,$$
$$h(y) \leq 0,$$

where $y = (x_{k+1},\ldots,x_N,u_k,u_{N-1})^T$,

$h(y) = (h_k(x_k,u_k),\ldots,h_{N-1}(x_{N-1},u_{N-1}))$

$f(y) = R_k(x_k,\ldots,u_{N-1})$,

$b = (c_k+V_k(x_k),c_{k+1},\ldots,c_{N-1})^T$,

(T denotes the transposition),

$$M = \begin{pmatrix} E_{k+1} & O & \ldots & O & -W_k & O & \ldots & O \\ -V_{k+1} & E_{k+2} & \ldots & O & O & -W_{k+1} & \ldots & O \\ & & \ldots & & & \ldots & & \\ O & O & \ldots & -V_{N-1} & E_N & O & O & \ldots & -W_{N-1} \end{pmatrix}$$

(E_i denotes the unit matrix of demension of X_i).

Observe that the Lagrangian function associated with this problem is the same as L_k. Using this fact and taking conditions (i),(ii) and (iii) into account we are now able to apply the Lagrange multipler theorem of convex programming (see Luenberg (1969), p.217) to the problem and this makes the proof complete. #

4. CONJUGATE FUNCTIONS

In this section we maintain all the assumptions made in Proposition 3.5. Following the method of conjugate functions, first we define

$$\rho_k(x_k,x,u,w,v) = R_k(x_k,x,u)$$

if $\quad x_{i+1} = g_i(x_i,u_i) + v_i,$

and $\quad h_i(x_i,u_i) \leq w_i, \quad i = k,\ldots,N-1;$

$$\rho_k(x_k,x,u,w,v) = +\infty \qquad \text{otherwise,}$$

and its conjugate function

$$\rho_k(x_k,t,s,\lambda,\mu) = \sup\{\langle t,x\rangle+\langle s,u\rangle+\langle w,\lambda\rangle+\langle v,\mu\rangle -$$

$$- \rho_k(x_k,x,u,w,v)\colon x,u,w,v\}.$$

The primal and dual problems will be

$$\min\{\rho_k(x_k,x,u,0,0)\colon x,u\} \qquad (P)$$

$$\max\{-\rho_k^*(x_k,0,0,\lambda,\mu)\colon \lambda,\mu\}. \qquad (D)$$

It is easy to see that (P) is the same as P(k), and in the case solutions for ρ_k^* exist, (D) is the same as D(k). We have similar results:

i) $-\rho_k^*(x_k,0,0,\lambda,\mu) \leq \rho_k(x_k,x,u.0,0)$ for each x,u,λ,μ

ii) If $-\rho_k^*(x_k,0,0,\lambda,\mu) = \rho_k(x_k,x,u,0,0)$

for some x,u,λ,μ, then (x,u) and (λ,μ) are optimal solutions of (P) and (D), respectively.

Now we define the primal and dual perturbation functions Φ_k, Ψ_k associated with ρ_k and ρ_k^* as follows:

(7)

$$\Phi_k(x_k,w,v) = \inf\{\rho_k(x_k,x,u,w,v): x,u\}$$

$$\Psi_k(x_k,t,s) = \inf\{\rho_k^*(x_k,t,s,\lambda,\mu): \lambda,\mu\}. \tag{7}$$

Proposition 4.1 Under the conditions of Proposition 3.5, Φ_k is a convex function of (w,v).

Proof. A direct verification will yield the proposition. #

Proposition 4.2 Assume additionally that (7) is solvable, then $\Phi_k,\ldots,\Phi_{N-1}$ yield the following recurrence relations

$$\Phi_i(x_i,w,v) = \inf\{f_i(x_i,u_i) + \Phi_{i+1}(g_i(x_i,u_i)+v_i,w',v'):$$

$$x_i,u_i \text{ with } h_i(x_i,u_i) \leq w_i\}$$

$i = k,\ldots,N-i$ and $\Phi_N(x_N) = f_N(x_N)$.

Proof. As the proof of this proposition is similar to that of Proposition 3.3, we omit it. #

Remember that problem (P) is said to be stable if the subdifferential $\partial\Phi_k(x_k,0,0)$ is a nonempty set. Now we can apply Theorem 5.11 (Avriel (1976)) to get the following result:

Let $\Phi_k(x_k,0,0)$ be finite. Then problem (D) has an optimal solution $(\bar{\partial},\bar{\mu})$ and

$$\Phi_k(x_k,0,0) = \max\{-\rho_k(x_k,0,0,\lambda,\mu): \lambda,\partial\}$$

$$= -\rho_k^*(x_k,0,0,\bar{\lambda},\bar{\mu}) \tag{8}$$

if and onyl if (P) is stable. Moreover, $(\bar{\lambda},\bar{\mu}) \in \partial\Phi_k(x_k,0,0)$

if and only if (8) holds.

The following results are immediate:

Corollary 4.1 Assume that (P) has an optimal solution (x,u). Then there exists (λ,μ) such that (x,u,λ,μ) is a saddlepoint for L_k if and only if (P) is stable. #

Corollary 4.2 Suppose that (7) is solvable. If P(k) is stable, then so are $P(k+1),\ldots,P(N)$ (these problems are determined by $x_{k+1},\ldots,x_N$ where $(x_{k+1},\ldots,x_N)$ together with some u is a solution of (7)). #

CONCLUSIONS

To conclude this paper we should emphasize that the results obtained are merely theoretical aspects of a duality approach for solving the dynamic problem described in Section 2. They provide us with a possibility of solving problem

$$\max\{d(x_i,\lambda,\mu): \ \lambda,\mu \text{ without constraints}\}$$

instead of

$$\min\{R_1(x_i,x,u): \text{ under constraints } (2),(3),(4)\},$$

where $d(x_i,\lambda,\mu)$ may be calculated by recurrence relations given in Corollary 3.1.

REFERENCES

Avriel, M., 1976, Nonlinear Programming (London: Prentice-Hall, INC.). Bellman, R., 1957, Dynamic Programming (New Yersey: Princeton University Press).

Evers, J.J.M., 1983, A Duality Theory for Infinite-horizon Optimization of Concave Input/Output Processes, Mathematics of Operations Research 8, 479-497.

Hadley, G., 1962, Linear Programming (London: Addision-Wesley).

Haneveld, W.K.K., 1985, Duality in Stochastic linear and dynamic programming (Amsterdam: Centrum voor Wiskunde en Informatica),

Luenberg, D.G., 1969, Optimization by Vector Space Methods (New York: John Wiley and Sons).

Weitzman, M.L., 1973, Duality Theory for Infinite-horizon Convex Models, Management Science 19, 783-789.

Dualitás a dinamikus programozásban

Dinh The Luc

Összefoglaló

A szerző véges-állapotu /"finite stage"/ dinamikus rendszerekre vonatkozó dualitás-elméletet dolgoz ki, felhasználva a matematikai programozásban használatos Lagrange és konjugált függvény módszert. Néhány dualitás tételt és nyereg-pont tételt is nyer, valamint rekurrencia-összefüggéseket, amelyek a rendszer dinamikus tulajdonságait jellemzik.

Двойственность в динамическом программировании

Дхин Тхе Лук

Р е з ю м е

В статье разработана теория двойственности для динамических систем конечного состояния /"finite-stage"/, используя метод двойственных функций Лагранжа в математическом программировании. Доказано несколько теорем двойственности и теорем о седловой точке, а также несколько рекуррентных соотношений характеризующих динамические свойства системы.

MTA SZTAKI Közlemények, 35/1986 p.105-125

ЛОРКА - РАСШИРЕНИЕ ПАСКАЛЯ К РЕЛЯЦИОННОЙ МОДЕЛИ БАЗ ДАННЫХ

М. Катриб Мора, Е. Квесада Орозцо, Й. Баш Байард, Е. Оберт Вазквез

Кафедра вычислительной техники, Гаванский Университет, Куба

ВВЕДЕНИЕ

Файлы являются единственным средством, предоставляемым языком ПАСКАЛЬ для хранения внешней по отношению к программам и, следовательно, годной для использования после окончания их выполнения, информации. Этого недостаточно для работы систем управления базами данных /СУБД/, т.к. любой программе, использующей файл, созданный другой программой, необходимо знать точное определение типа данных, содержащихся в нем, о чем ПАСКАЛЬ не записывает никаких сведений. Любое изменение в структуре файла требует перепрограммирования и перекомпиляцию всех программ, его использующих.

С другой стороны, организация и управление внешней памятью осуществляется операционной системой /ОС/, под управлением которой работает компилятор с языка высокого уровня /будь это ПАСКАЛЬ или любой другой/, что вынуждает СУБД, ориентированные на использование файлов, ограничиваться возможностями ОС. По этой причине очень часто в таких системах информация, содержащаяся в базах данных, не защищена и доступна "чужим" программам.

Компонентны таких файлов должны иметь, как правило, одинаковые размеры. Этот факт ограничивает более динамическое использование памяти, поскольку не все виды информации, встречающиеся в приложениях, имеют такие характеристики.

ЧТО ТАКОЕ ЛОРКА

ЛОРКА - это СУБД, основанная на модульном расширении ПАСКАЛЯ. Это расширение состоит в добавлении новых типов, переменных, процедур и функций, составляющих удобный интерфейс для работы

с понятиями реляционной модели и в то же время позволяющих скрыть от пользователя особенности работы системы.

ЛОРКА предоставляет пользователю:

- Широкий набор типов данных, облегчающих создание баз данных /БД/, особенно в области научных исследований.
- Гибкие интерактивные средства для ввода и модификации данных посредством экранных редакторов.
- Быструю селективную выборку информации, хранящейся в БД, на основе паскалевидных формул, обобщающих нотацию реляционной алгебры.
- Возможность хранения этих формул внутри самой БД, чтобы они могли быть применены без необходимости повторного ввода.
- Возможность организации отношений, что гарантирует доступ к любой отдельной записи.
- Большую экономию памяти на основе использования различных динамических форм представления информации.
- Возможность быстрых и надежных копий БД и передачи информации между ними.
- Возможность взаимодействия с файловой системой ОС и передачи информации от Паскаль-файлов в БД и обратно.

ТИПЫ ДАННЫХ

Множество возможных значений любого атрибута, а также допустимые над ними операции характеризуют то, что будем называть типом данных /ТД/ атрибута.

Типы данных в ЛОРКе разделяются на: простые, текстовые, кодовые и структурные. Простыми типами являются: целый, байтовый, вещественный, а также цепочки, даты и литералы. Структурными ТД являются векторы и множества элементов любого простого типа.

ЦЕЛЫЙ Т.Д.

Целый Т.Д. соответсвует типу integer языка ПАСКАЛЬ.

БАЙТОВЫЙ Т.Д.

Это тип отвечает определению:

```
type byte=0..255
```

ВЕЩЕСТВЕННЫЙ Т.Д.

Использование вещественных чисел в формате с плавающей запятой характерно для всех научных применений. Включение в ЛОРКу числовых данных в виде целых и вещественных величин позволяет использовать систему в области научных исследований, а также достичь более компактную форму представления данных в целях экономии памяти. Этот тип подобен стандартному типу real языка ПАСКАЛЬ.

ДАТЫ

ЛОРКА включает предопределенный тип дата. Константа этого типа записывается в виде:

<day> : <month> : <year>

где <day> обозначает целую константу в ранге 1.31, <month> целую константу в ранге 1.12, и <year> - положительное целое.

П р и м е р ы :

12 : 3 : 78 30:11:84 1:1:1985 12:3:1978

Соответствующий тип языка ПАСКАЛЬ определяется как:

```
date = packed record
         day   : 0 .. 255;
         month : 0 .. 255;
         year  : 0 .. maxint
       end
```

ЛИТЕРАЛЫ

Во многих приложениях необходимо манипулировать атрибутами, значения которых отображают качественными характеристиками, неподдающимися количественному измерению.

В большинстве существующих СУБД, этот тип данных представляет-

ся в виде цепочек символов /что является неэкономичным, поскольку должна выделяться память для целой цепочки при каждом появлении одного и того же значения/, либо представляется в виде чисел, что заставляет пользователя заботиться о кодировании и декодировании этих переменных.

ЛОРКА представляет пользователю тип ЛИТЕРАЛ. Константа этого ТД может быть цепочка символов /максимальная длина которой определяется при создании БД/. Значения атрибутов, отвечающих этому ТД, задаются пользователем в виде цепочек, а предсталяются они внутри системы кодированным способом, что приводит к большей экономии памяти и эффективности во времени.

ЛОРКА также включает предопределенное значение НЕИЗВЕСТНОЕ, которое воспринимается по умолчанию как значение любого атрибута этого ТД, который не был инициализирован явным образом.

> Перечисленные типы Паскаля являются недостаточными для того, чтобы отражать суть этого понятия, т.к. Паскаль не сохраняет в памяти перечисленные идентификаторы.

В Паскаль-расширении этот тип должен считаться частным и может быть использован только через собственные ресурсы расширения:

```
type literal = private
```

ЛОРКА располагает, кроме того, двумя процедурами для преобразования цепочек языка Паскаль в литералы и наоборот.

```
function Str_to_lit(lit_name:string;
                    var lit:literal):boolean;
procedure Lit_to_str(lit:literal;
                    var lit_name:string)
```

В отличии от перечисляемых типов Паскаля, включение литералов в БД может осуществляться динамическим образом с помощью процедуры:

```
procedure Insert_lit(lit_name:string)
```

Литералы можно переименовать с помощью процедуры:

```
procedure Rename_lit(old_name,new_name:string)
```

Процедура

```
procedure Literals_to_file(file_name:string)
```

создает текстовый файл, содержащий имена всех литералов любой БД.

ЦЕПОЧКИ

Для тех атрибутов, диапазон значений которых должен отображаться последовательностью символов и которые не могут быть представлены литералами, т.к. невозможно перечислить совокупность всех возможных значений, в ЛОРКу включен тип данных string.

Этот ТД подобен цепочечному типу языка Паскаль. Однако в ЛОРКе значения этого типа хранятся внутри БД согласно своей текущей длине, а не какой-то максимальной.

ВЕКТОРНЫЕ СТРУКТУРЫ

Во многих научных приложениях, например, для математико-статистической обработки, необходимо манипулировать данными, предоставленными в виде векторов вещественных чисел. Однако, в реляционной модели БД эта форма представления неэффективна, поскольку заставляет пользователя задавать имя каждому из атрибутов.

В ЛОРКу включена возможность определить векторы из элементов, принадлежащих любому из простых ТД /кроме цепочечного/. Длина вектора определяется пользователем при создании БД.

Таким образом, оперируя одним атрибутом /и одним именем/ можно иметь доступ к целому вектору. Когда требуется обращение к конкретному элементу вектора, этого можно добиться с помощью индекса, как это делается в большинстве языков программирования.

Значение типа ВЕКТОР представляется в системе в виде:

$$[\langle value_1 \rangle, \langle value_2 \rangle, \ldots \langle value_n \rangle]$$

где каждое value обозначает значение простого типа. Соответствие между структурными типами этого класса ЛОРКи и Паскаля следующее:

```
     array[1..n] of integer   - для целочисленных векторов
     array[1..n] of real      - для векторов вещественных чисел
  packed array[1..n] of byte  - для байтовых векторов
  array [1..n] of literal     - для векторов, составленных из
                                литералов.
```

СТРУКТУРА ТИПА МНОЖЕСТВА

В ЛОРКу включены в качестве типов атрибутов множества, состоящих из значений любого из простых типов. Атрибут, определенный таким образом, может иметь своим значением множество значений базового типа.

Любое значение этого типа обозначается в форме:

$$\{<value_1>, <value_2>, \ldots, <value_n>\}$$

где каждое value обозначает значение простого типа.

Множество в ЛОРКе не имеют эвивалента на ПАСКАЛе. Работа со значениями этого типа может осуществляться только посредством функций и процедур, предоставленных системой.

ТЕКСТОВЫЙ ТИП

Этот ТД используется для представления в БД текстов большой длины, такие как документы или резюме документов. Значения атрибутов текстового типа могут быть показаны на дисплее для модификации с помощью экранного редактора.

Текстовый тип может быть использован также для представления формул реляционного исчисления. Таким образом, пользователь может хранить свои формулы внутри собственной БД, чтобы они могли быть использованы и/или модифицированы в любой момент.

Этот ТД не имеет аналога на языке ПАСКАЛЬ и дожен считаться частным /private/. По этой причине его можно использовать толь-

ко через системные функции.

```
type db_text = private.
```

КОДОВЫЙ ТИП

Результат компиляции текста, соответствующего формуле реляционной алгебры, является значением кодового типа. Такие значения не могут быть распечатаны, т.к. они соответствуют внутреннему представлению формулы отбора.

Хотя ЛОРКА представляет общую интерактивную версию, текстовые и кодовые значения используются для разработки новых версий.

ОПЕРАЦИИ С БД

Система располагает следующими ресурсами для работы с БД:

а/ Создание БД

Эту операцию можно выполнить с помощью

```
procedure Define_DB(db_name:string;
                    protection_key:string;
                    size:0..maxint);
```

При этом определяется пустая БД: без отношений и с нулевым словарем.

б/ Открытие БД

```
procedure Use_DB(db_name:string;
                 protection_key:string)
```

Открывает для последующего использования базу данных. Все операции, которые будут рассматриваться дальше, относятся к открытой БД и поэтому не делают явную ссылку к ней.

Для того, чтобы узнать количество свободной памяти в БД используется:

```
function Mem_in_DB:integer;
```

которая возвращает колическо свободных страниц /1 страница = 1024 байтов/.

ОТНОШЕНИЯ В ЛОРКА-ПАСКАЛе

Отношения представляются в ЛОРКе с помощью типа.

Этот тип не имеет эквивалента на ПАСКАЛе и поэтому с объектами такой натуры можно работать только через возможности ЛОРКи.

```
type relation = private
```

Отношения могут находиться в одном из двух состояний или режимов работы: query - если над ней не будут выполняться никаких преобразований /информация будет только считываться/, или actualization, если будут производиться модификации.

```
rel_operation = (query, actualization)
```

ОПРЕДЕЛЕНИЕ ОТНОШЕНИЯ

```
procedure Define_rel(rel_name:string)
```

Для определения реляционной схемы отношения необходимо определить его атрибуты. Это делается с помощью процедур Define_atrib,Append_atrib Copy_structure. Для того, чтобы охарактеризовать типы значений атрибутов задаются:

```
type Data_type = (Int_type,Byte_type,Real_type,Literal_type,
                  Date_type,String_type,Text_type,Code_type,
                  Int_vector,Byte_vector,Real_vector,
                  Date_vector,Literal_vector,
                  Int_set,Byte_set,Real_set,Literal_set,
                  Date_set,String_set);
type atribute = private
```

Тип надо рассматривать как собственный тип ЛОРКи и может быть использован только через следующие операции:

а/
```
procedure Define_atrib(rel_name:string;
                       atrib_name:string;
                       atrib_type:data_type;
                       length_vector:0..maxint)
```
включает атрибут в схему отношения.

б/ Если пользователь желает добавить в схему атрибуты, уже

существующие /определенные в других отношениях/, используются:

```
procedure Append_atrib(rel_name:string;
                       atrib_name:string)
```

в/
```
function Str_to_atrib(atrib_name:string;
                      rel_name:string;
                      var atrib:atribute):boolean
```

Эта функция определяет, является ли atrib_name именем атрибута данного отношения и возвращает этот атрибут в переменной atrib.

г/ Тип значений атрибута может определяться функцией:

```
function atrib_type(atrib:atribute):data_type
```

д/ Имя атрибута можно определить процедурой:

```
procedure Atrib_to_str(atrib:atribute;
                       var atrib_name:string)
```

е/ Операция

```
procedure Rename_atrib(old_name:string;
                       new_name:string)
```

позволяет модифицировать имя атрибута.

ж/
```
procedure Copy_structure(source_rel_name,
                         destination_ret_name:string)
```

Добавляет к реляционной схеме второго отношения те атрибуты первого /исходного/ отношения, не являющиеся атрибутами второго отношения.

ОБЩИЕ ОПЕРАЦИИ НАД ОТНОШЕНИЯМИ

ЛОРКА включает операции, действующие одновременно над целым отношением:

а/
```
function Delete_rel(rel_name:string):boolean
```

Эта функция стирает /убирает/ отношение из БД и освобожда-

ет им занятую память.

б/ function Sort_rel(rel_name:string;
atrib:atribute):boolean

Производит сортировку отношения по заданному атрибуту.

в/ procedure Tuples_rel(rel_name:string;
var total_tuples:integer)

Возвращает количество кортежей /записей/ названного отношения.

г/ function Total_atribs(rel_name:string):integer

Возвращает общее количество атрибутов отношения/в случае ошибки возвращает нуль/.

д/ procedure Copy_rel(source_rel_name,
destination_rel_name:string)

Создает копию исходного отношения с именем второго отношения. Если раньше существовало отношение с именем destination rel_name, то прежнее его содержимое уничтожается.

е/ procedure Union_rels(rel1_name,rel2_name:string)

Создает новое отношение, являющееся результатом соединения отношений rel1_name, rel2_name.

ж/ procedure Union_rels(rel1,name,rel2_name:string)

Оба отношения должны иметь одинаковую реляционную схему. Эта процедура создает отношение, являющееся объединением rel1_name, rel2_name.

з/ procedure Concat_rels(rel1_name,rel2_name:string)

Оба отношения должны иметь одинаковую реляционную схему. Создается отношение, являющееся результатом добавления /т.е. могут повторяться кортежи/ к отношених rel1_name кортежи отношения rel2_name.

и/ Процедура
procedure Rename_rel(rel_name,new_name:string)
позволяет переименовать отношение.

ОПЕРАЦИИ ДЛЯ ОБРАБОТКИ И МОДИФИКАЦИИ ОТНОШЕНИЙ

а/ Открытие отношения

Операции, о которых будет говориться дальше, позволяют обработать данные, содержащиеся в любой БД, с программ, работающих в системе.

```
procedure Open_rel(rel_name:string; mode:rel_operation;
                   var rel:relation)
```

связывает имя отношения со значением типа relation, который в дальнейшем будет использоваться при работе с отношением.

Если отношение открывается в режиме query, информация, имеющаяся в отношении, может быть только считана; если отношение открывается в режиме modification, то информация готова как для считывания, так и для записи /модификации/.

б/ Текущий кортеж

В любой момент над каждым открытым отношением возможен доступ только к одному кортежу, называющемуся текущим. Таким образом, обработка отношения производится путем просматривания его с помощью текущего кортежа.

После открытия отношения текущим является самый первый кортеж.

Процедура

```
procedure Get_tuple(rel:relation)
```

делает текущим следующий кортеж названного отношения.

Если его не существует, то булевая функция Eorel(rel) возвращает true, и назначение текущего кортежа становится неопределенным.

Функция

```
function Eorel(rel:relation):boolean
```

возвращает значение true, если задана Get_tuple при текущем кортеже, равным последнему кортежу отношения, или если открывается пустое отношение. В других случаях возвращает

false.

Процедура

procedure Reset_rel(rel:relation)

позиционирует уже открытое отношение в первом его кортеже.

в/ Прямой доуступ к кортежу

```
function Find_tuple(var Pascal_data;
                    rel:relation;
                    atrib:atribute):boolean
```

Ведет поиск, начиная с текущего кортежа, следующего кортежа отношения, имеющего значение атрибута atrib, равным параметру Pascal_data. Если кортеж существует, то он становится текущим и функция возвращает true; в противном случае изменений не происходит и функция возвращает false.

г/ Стирание кортежей

procedure Delete_tuple(rel:relation);

Применима только в случае, если отношение rel открыто в режиме actualization. Физически убирает текущий кортеж.

д/ Копирование кортежа

procedure Copy_tuple(sourece__el,destination_rel:relation)

Копирует текущий кортеж исходного отношения после текущего кортежа отношения назначения и делает его текущим. При этом реляционные схемы обоих отношений должны совпадать.

е/ Добавление кортежа

Вставляет нулевой кортеж /каждый его атрибут инициализируется нулевым значением соответствующего ТД/ после текущего кортежа отношения. Этот нулевой кортеж может быть потом отредактирован с помощью предоставленных системой возможностей. Если eorel имеет значение true, то нулевой кортеж становится текущим и eorel переводится в false.

ж/ Редактирование кортежей

```
procedure Edit_tuple(rel:relation; mode:edit_mode);
where edit_mode=(new_tuple, old_tuple)
```

При обращении к этой процедуре в режиме old_tuple показывается на дисплее текущий кортеж отношения и вызывается редактор кортежей ЛОРКи. Результат редактирования заменяет текущий кортеж /отношение должно быть открытым для модификации/. При обращении к ней с параметром new_tuple возможно диалоговое редактирование нового кортежа.

з/ Соединение кортежей

```
function Join_tuples(rel1,rel2:relation)
```

Возвращает true, если текущие кортежи отношений rel1, rel2 удовлетворяют усвловиям соединения.

```
procedure Append_tuples(rel1,rel2,rel3:relation)
```

Вставляет новый кортеж в rel3, что соответствует добавлению текущему кортежу rel1 значений атрибутов отношения rel2, которые не являются атрибутами rel1.

ПЕРЕДАЧА ЗНАЧЕНИЙ МЕЖДУ ЛОРКОЙ И ПАСКАЛЕМ

Следующие операции работают над текущим кортежом отношения и передают ПАСКАЛю значение определенного атрибута кортежа или модифицируют /паскалевой константой/ значение одного атрибута.

а/ Передача значений в Паскаль

```
procedure Get_value(rel:relation;
                    atrib:atribute;
                    var Pascal_data)
```

Возвращает через параметр Pascal_data соответствующее значение языка Паскаль.

В случае, когда атрибут является вектором, фактически параметр должен иметь соответствующий тип array языка Паскаль. Для того, чтобы иметь доступ к отдельным компонентам без необходимости передачи всего вектора включена операция:

```
procedure Get_item_vector(rel:relation;
                          atrib:atribute;
                          index_item:integer;
                          var Pascal_data)
```

которая возвращает компоненту index_item вектора. Для того, чтобы работать со значениями множественных атрибутов даются следующие операции:

1.
```
function Cardinal_set(rel:relation;
                      atrib:atribute):integer;
```

   Возвращает количество элементов множества с именем atrib текущего кортежа отношения rel.

2.
```
procedure Get_item_set(rel:relation;
                       atrib:atribute;
                       index_item:integer;
                       var Pascal_data)
```

   Возвращает в параметре Pascal_data элемент index_item множественного атрибута atrib отношения rel.

3.
```
function Memeber(rel:relation;
                 atrib:atribute;
                 var Pascal_data :boolean)
```

   Эта функция является предопределенной для достижения большей эффективности работы системы, т.к. она является основой всех множественных операций.

б/ Копирование значений одного атрибута в другой

```
procedure Copy_value(source_rel:relation;
                     source_atrib:atribute;
                     dest_rel:relation;
                     dest_atrib:atribute)
```

С помощью этой процедуры возможно формирование кортежей, на основе кортежей других /или даже того же/ отношений без необходимости передачи посредством Паскаль-значений.

в/ Передача значений из ПАСКАЛЯ в ЛОРКУ

Следующая операция является обратной по отношению к Get_value:

```
procedure Put_value(rel:relation;
                    atrib:atribute;
                    var Pascal_data)
```

Если атрибут есть вектор, тогда фактически параметр Pascal_data должен быть соответствующего векторного типа.

Для модификации элементов множества имеется операция

```
procedure Put_item_set(rel:relation;
                       atrib:atribute;
                       var Pascal_data)
```

которая включает элемент Pascal_data во множество значений атрибута atrib.

```
procedure Put_item_vector(rel:relation;
                          atrib:atribute;
                          index_item:integer;
                          var Pascal_data)
```

Присваивает значение Pascal_data в качестве index_item компоненты вектора значений атрибута atrib в текущем кортеже.

```
procedure Rem_item_set(rel:relation;
                       atrib:atribute;
                       var Pascal_data)
```

Убирает из множества значений атрибута atrib элемент, равным параметру Pascal_data.

ТЕКСТЫ В ЛОРКА-ПАСКАЛЕ

а/ Редактирование текстов

Создание и модификация атрибутов этого типа может выполняться только через

```
procedure Edit_text(source_text:db_text;
                    var dest_text:db_text;
                    mode:edit_mode)
```

которая соответствует экранному редактору ЛОРКИ.

Если программист больше не будет использовать значение текстового типа, то его ответственностью является освобождение запятой текстом памяти. С этой целью включена операция:

```
procedure Free_text(text_to_deallocate:db_text)
```

б/ Передача текстов между БД и файлами Паскаля

```
procedure File_to_text(source_file:string;
                       var dest_text:db_text)
```

source_file должен являться именем текстового файла операционной системы, под управлением которого работает Паскаль. Эта процедура передает содержимое файла в ЛОРКу.

```
procedure Text_to_file(source_text:db_text;
                       dest_file:string)
```

Передает текст source_text из ЛОРКи в текстовый файл.

Этим файлом могут являться тоже АЦПУ или дисплей.

КОДЫ В ЛОРКЕ

Текст, полученный с помощью предыдущих операций, может отражать формулу реляционной алгебры. Эта формула может быть преобразована к виду, более близкому к машинному языку с помощью

```
procedure Compile(text_formula:db_text;
                  var code_formula:code;
                  var OK:boolean)
```

Этот код генерируется в оперативной памяти. Для записи его в БД используется

```
procedure Put_code(icod:code; var cod:db_code)
```

Значения типа db_code могут быть включены в кортежи БД через соответствующее присваивание с помощью put_value. Для освобождения внутренней памяти, занятой кодовым значением используется процедура

```
procedure Free_code(cod:code);
```

Для освобождения внешней памяти используется

```
procedure Free_dbcode(cod:db_code)
```

Кроме того, ЛОРКА представляет процедуру garbage_collector, которая собирает всю занятую память, которая больше не будет использоваться.

ВЫПОЛНЕНИЕ КОДОВ

Результат компиляции формулы, т.е. кодовое значение, может быть применено к текущему кортежу одного /или двух, если запрос подразумевает соединение/ отношения с помощью

```
function Apply var(code_formula:db_code;
                   rel1,rel2:relation):boolean
```

Если кортеж /кортежи/ удовлетворяет /ют/ условию, функция возвращает true.

Если пользователь желает выполнить ранее скомпилированный запрос, который хранится вБД в виде значения db_code, это значение необходимо загрузить в оперативную память с помощью

```
procedure Load(ext_cod:db_code;
               var internal_code:code)
```

Включение в ЛОРКА-ПАСКАЛЬ операций edit_text, compile, apply делает возможным создание и применение запросов во время выполнения прграммы без необходимости их явного включения в структуру программы, что приводит к высокой степени независимости и гибкости. Таким образом, можно хранить даже внутри самой БД запросы /в форме текстов и/или кодов/ и задать их в течение времени прогона любой программы.

Пример

Допустим в БД научно-технической информации имеется отношение USERS /Пользователи/ с атрибутами.

```
Name(string)
Dpt (literal)
Key_wards (db_code)
```

и имеется отношение PAPERS /Документы/ с атрибутами

```
Title   (string)
Author  (string)
Date    (date)
Journal  (literal)
Subject  (set of literals)
```

Атрибут key_words отражает профиль библиографических интересов пользователя, например:

```
With PAPERS do(Journal = CACM) and
{PASCAL,ADA} disjoint Subject
```

который выбирает те кортежи отношения PAPERS из журнала CACM, имеющие тематикой любое из двух значений PASCAL или ADA.

Если бы мы хотели распечатать названия работ, интересующих преподавателей департамента Вычислительной Техники /Computer Sciences/, то можно было бы поступить так:

```
var dept,ComputerSciences:literal;
    key_words_atrip,title_atrib,dept_atrib:atribute;
    atrib_type:data_type;
    user_kw:db_code; cod_user_kw:code;
    title:string;
    null_rel,users_rel,papers_rel:relation;

begin
  str_to_atrib(d'dpt','USERS',dept_atrib,atrib_type);
  str_to_atrib('key_words','USERS',key_words_atrib,atrib_type);
  str_to_atrib('title','PAPERS',title_atrib,atrib_type);
  if not st_to_lit('ComputerSciences',ComputerSciences) then
    writeln('Error ComputerSciences must be a literal');
  open_rel ( 'USERS',query,users_rel);
  open_rel ( 'PAPERS',query,papers_rel);
  while not eorel(users_rel) do
```

```
begin
  get_value(users_rel,dept_atrib,dept);
   { Get user department }
  if dept=ComputerSciences then
    begin
      get_value(user_kw,users_rel,key_words_atrib);
       { Get user key words interests }
      load(user_kw,cod_user_kw):
      while not eorel(papers_rel) do
       begin
         if apply cod user kw,papers_rel,null_rel) then
          begin
            get value(title,papers_rel,title_atrib);
              { Get a title from a paper }
            writen(title)
          end;
         get_tuple(papers_rel)  { Get new paper }
       end;
      free_code(cod_user kw);
      reset(papers_rel)
    end;
  get_tuple(user_rel) { Get new user }
 end
end
```

ЗАКЛЮЧЕНИЕ

Интерактивная версия ЛОРКИ разработана, используя вышеописанное расширение Паскаля, что облегчает ее использование, если требуется только классические операции реляционного исчисления. Включение различных предопределенных типов данных и возможности создания и обработки баз данных с языка ПАСКАЛЬ создает благоприятные условия для разработки конкретных рабочих сред.

Включение цепочек перемнной длины, литералов и множеств обеспечиваеют большую гибкость использования и экономию памяти по сравнению с другими системами, работающими на основе полей формированной длины.

Описанные возможности не представляются пользователям в других СУБД, имеющих узкоспециализированные языки, очень бедные по сравнению с современными языками программирования как Pascal.

Так как ПАСКАЛЬ не дает никаких механизмов для определения частных типов, настоящее расширение требует дисциплинированного применения со стороны программистов, которые должны использовать эти ТД только через предоставленные функции и процедуры. Препроцессор или новый компилятор могут являться будущим решением.

LORKA: a PASCAL-nak egy kiterjesztése relációs adat-bázis-kezelő rendszerek számára

M. Katrib Mora, E. Quesada Orozco, J. Bosch Bayard,

E. Aubert Vazquez

Összefoglaló

A LORKA egy relációs adat-báziskezelő rendszer, amely a PASCAL nyelvnek egy moduláris kibővitésén alapszik. A kibővités során uj tipusok, változók, eljárások és függvények jöttek létre. A kibővités egy olyan interfejsz--nek tekinthető, amely segitségével jól lehet dolgozni a relációs adat-bázis fogalmaival anélkül, hogy a felhasználót a rendszer müködésének részleteivel terhelné.

LORKA: an extension of PASCAL for the relational data-base models

M. Katrib Mora, E. Quesada Orozco, J. Bosch Bayard,

E. Aubert Vazquez

Summary

LORKA is a system for treating the relational data-base models that is based on a modular extension of PASCAL. In the course of extension new types, variables, procedures and function have been created. The extension can be considered as a good interface enabling good work with the basic notions of relational data-base models but at the same time the user is not burdened by the details of the work of the system itself.

MTA SZTAKI Közlemények, 35/1986 p. 127-134

DATA BASE MANAGEMENT SYSTEM dBASE-300

FOR CUBAN MINICOMPUTER CID 300/10

Lic. MIGUEL FONFRÍA ATÁN
Lic. EUGENIA MUÑIZ LODOS
Lic. Ma ELENA BRAGADO BRETANA
Lic. LUIS FAJARDO ÁLVAREZ de la CAMPA
Lic. JORGE L. de la CANTERA RUIZ
Lic. LUIS E. FERNÁNDEZ LARA
Tech. SILVIA PÉREZ MARTÍNEZ

IMACC, Academia De Ciencias
Habana, Cuba

1. INTRODUCTION

During the late seventies and early eighties of this century, a host of Data Base Management Systems (DBMS) have emerged, based on the Relational model presented by E. F. Codd in 1970. These microcomputer-implemented DBMS have spread this technology everywhere, although, nearly all of them fail to abide by the principles of Data Base theory. However, these systems provide simple means for the development of applications with a certain degree of data--program independence, means to reduce duplication of information, and also a language for data treatment with a high level and available to the average user.

As a result of this situation, as a complement to the software devised for CID 300/10 (similar to SM-3) a DBMS was developed to carry through the following purposes:

- easy use language,
- efficient management of data structures according to the hardware's memory restrictions,
- to ensure work with the rest of the software, and specially with Operating System for Commercial applications GES-300.

An early stage of the work was the assessment of the various DBMS available in order to determine whether it was necessary to design a new data management language and learn the general characteristics of present systems with a view to incorporate these into our system. From this study the conclusion was made that:

- a wide range or Relational DBMS exists in the market, each with a different man-machine interphase,
- one of the DBMS, the dBASE, is almost a standard for microcomputers due to its extended use in our country, its facilities and its simple design,
- the deficiencies set before the dBASE in respect of the rate of processing could be removed in the CID 300/10.

We accordingly set ourselves the aim of implementing a DBMS which from the user's point of view should be compatible with dBASE.

The dBASE-300 contemplates files used to be compatible with relative organization files of GES-300's COBOL, so that continuity of the work is ensured with applications previously developed for CID 300/10.

Now we shall expound the general characteristics of dBASE-300, pointing out the major techniques used in its development as well as the main differences regarding dBASE II of microcomputers.

2. General characteristics of the system

As DBMS standard the general features of dBASE II, version 2.4, were taken, developed for 8 bit microcomputers. Also

enciosed were some of the facilities of dBASE III which considerably increase the system's potentials. These are:

- the possibility to use more than 2 files simultaneously,
- precision of the arithmetic up to 18 digits,
- file classification by up to 6 keys simultaneously,
- command AVERAGE,
- some of the functions which are not in dBASE II.

Development of dBASE-300 has been carried out in FOBOS's macro-assembler language.

Implementation of the DBMS dBASE-300 is logically divided into one compiler and one command executer.

To implement the compiler a LALR/1/ grammar was designed that contemplates the required commands.

This grammar was processed by the table generator for syntactical analysis of LALR/1/ grammars of CID 201-B.

Due to capacity problems, this program could not process the whole grammar. Therefore, the syntactical analysis was carried out by using the two methods together: automated, which provided us with the table generator, and ad hoc in the portion that could not be processed by the generator. This was the more laborious part of the project on account of implementation of these two form of work, and also because the table generator of CID 201-B affords only very scanty facilities for its use.

Inside the executor, a module was designed that performs dynamic allocation of memory, so that any process needing memory asks for it. This module is based on a FIRST-FIT method and ensures compaction of free memory, if needed.

Also in the executor there is a module in charge of every input-output processes of files by solving both the sequential and the indexed organisation, for the latter purpose using the B+ tree structures for dense indexes.

The executor for command work has a decimal arithmetic with a capacity of up to 18 digits. This module also solves all arithmetical, chain and logical functions provided by dBASE-300.

A module has been included inside the executor for treatment of the terminal. This module responds to the heavy requirements of commands using this input-output medium.

At the time of execution, syntactical analysis of expressions supplied by the user is made using the recursive descendent method. This occurs in commands such as INPUT, ACCEPT and REPORT.

The potential of command REPORT /report generator/ of dBASE-300 is increased with the possibility of defining up to 6 breakage fields.

To provide for programming of the classification command the SHELL method, which had already reported good results in the utility programs of CID 300/10, was selected, and a treatment of the auxiliary files with two balanced buffers was added. A considerable improvement of execution times was achieved.

3. Main differences from microcomputers' dBASE II

3.1 In the instruction formats, where reference is made to a file name, a specification of FOBOS files must be supplied instead of type CP/M or PC-DOS specifications. In creation /CREATE/, if a variable size data base is to be created a maximum quantity of blocks should be given, using option /n/ of the FOBOS specification.

3.2 The number of open files depends on the free memory left by the System Generation /at present 5 files/. Several files being open at a given time, it is possible to indistinctly make reference to any of the files' fields and no additional procedure is required. To ensure this one has to bear in mind that field names should not be repeated in different files.

3.3 The records in the data files should have a length of 4 or more characters, up to a maximum of 512. The number of fields in a file is determined by character length of the field name, that is, the shorter the field name, the higher

the number of fields in a file. If, for instance, we have as an average 6 characters in the field names, 32 fields can be defined in a file.

3.4 Besides the dBASE II functions, dBASE-300 has the TIME(), MONTH(), CMONTH, CDATE, and DATE functions.

3.5 Command syntaxis in dBASE-300 is rigid, unlike that in dBASE II, where writing

DISPLAY ALL CAMPO FOR AUX=100

is allowed, which is equivalent to:

DISPLAY FOR AUX=100 ALL CAMPO.

This is not allowed in dBASE-300, where the order of commands expressed in dBASE II's manual must be adhered to. Nor is a reduction of command or option names to 4 letters allowed; that is, for instance, SELECT is not equivalent to SELE.

3.6 Keys CTL-S and CTL-Q having already been used by FOBOS Operation System, keys CTL-H and CTL-X are substituted, respectively.

3.7 It being possible to have more than 2 files open simultaneously in dBASE-300, SELECT instruction's format is now changed to SELECT <file> , where <file> is a file that was previously a parameter of a USE instruction.

3.8 The following are not implemented:

SET STEP ON/OFF	SET INTENSITY ON/OFF
SET ALTERNATIVE ON/OFF	SET DEBUG ON/OFF
SET LINKAGE ON/OFF	SET CARRY ON/OFF
SET BELL ON/OFF	SET RAW ON/OFF
SET ESCAPE ON/OFF	SET ALTERNATE TO
	SET INDEX TO

3.9 In dBASE-300, character # is not allowed in the conditional commands. < > must be substituted.

In order to make automatic the transfer of microcomputer applications to CID 300/10, programs are being made up that change both the microcomputer programs and data bases of dBASE II to dBASE-300. Besides, utility programs, such as ZIP, are being implemented to facilitate operation with the system.

Documentation of dBASE-300 is the same as that of dBASE II.
An annex is added which details the above differences.

4. CONCLUSIONS

With the acquisition of a Data Base Management System /DBMS/ for System CID 300/10, the user is provided with technology for data processing. Complemented with the other utilities of software, this system enables fast information retrieval from a data base created, maintained and controlled by different program packages already implemented. It enables the development of applications, with a high degree of data-program independence resulting in a saving on peaking and maintenance. It also enables exchanging application programs between microcomputers and CID 300/10, thanks to language compatibility.

The primary aims of this undertaking have been achieved with high quality level according to the results that were sought.

This work has been helpful in raising the technical level of those who use System CID 300/10 and at the same time has been instrumental in updating and improving the makers' technical level.

REFERENCES

1. Codd, E. F. A relational model for large shared data banks. Comm ACM 13:377-387, 1970.

2. Békéssy, A., and J. Demetrovics. Contribution to the theory of data base relations. Discrete Mathematics 27:1-10, 1979.

3. Date, C. J. An introduction to Data Base Systems. Dirección de Cálculo Electrónico, JUCEPLAN.

4. Demetrovics, J., E. Knuth, and P. Radó. Specification Meta Systems. IEEE, May 1982.

5. Knuth, D.E. The Art of Computer Programming, volume I: Fundamental Algorithms. Addison-Wesley.

6. Martin, James. Data Base Organisation. Edi. Prentice/Hall International.

7. Ratliff, Wayne. dBASE User Manual. Ratliff Software Production.

dBASE-300: egy adatkezelési rendszer a CID 300/10 kubai miniszámitógépre

M.F. Atan, E.M. Lodos, E.B. Bretana, L.F.a. CAMPA,
J.L.C. RUIZ, L.E.F. LARA, S.P. MARTINEZ

Összefoglaló

A dBASE-300 rendszert a következő célok elérésére tervezték: könnyü leiró-nyelv; a hardverhez alkalmazkodó adat-kezelés; kompatibilis a GES-300 operációs rendszerrel. A cikkben a dBASE-300 általános jellemzése szerepel, valamint megirásának fő technikája.
A cikk a dBASE-300 és a forgalomban levő más adatkezelő rendszerek fő különbségeire is kitér.

dBASE-300: система манипуляции базами данных для кубинского мини-компьютера CID 300/10

М. Фонфриа Атан, Е. Муниз Лодос, Е. Брагадо Бретана,
Л.Ф. Алварез де ла С., Й.Л. де ла Кантера Руиз, Л.Ф. Фернандез Лара, С. Перез Мартинез

Р е з ю м е

Система dBASE-300 была построена для следующих целей: простой язык; приспособленность к хардверу; компатибельность с операционной системой GES-300. В статье описаны базисные черты системы и техника ее построения. Статья также касается главных различий между dBASE-300 и остальными похожими системами.

MTA SZTAKI Közlemények 35/1986 p. 135-146

ОЦЕНКА ЭФФЕКТИВНОСТИ КОМБИНАТОРНОГО АЛГОРИТМА РАСПОЗНАВАНИЯ ОБРАЗОВ

Нго Куок Тао, Хоанг Кием

Институт информатики и кибернетики
Ханой - Вьетнам

В этой статье будет показана оценка ошибочной вероятности комбинаторного алгоритма распознавания образов для N исходных алгоритмов и утверждено, что если каждый исходный алгоритм имеет ошибочную вероятность $0 < \varepsilon < 1/2$, то

$$R_N = \sum_{k>\lfloor N/2 \rfloor} \binom{N}{k} \varepsilon^k (1-\varepsilon)^{N-k} \leq \exp(-N.c(\varepsilon)) \qquad /1/$$

где $\lfloor x \rfloor$ - наибольшее целое число, не превосходящее x,

$$c(\varepsilon) = \ln\left(\frac{1}{2\sqrt{\varepsilon(1-\varepsilon)}}\right) > 0 \ .$$

Более того, $\forall \varepsilon \quad 0 < \varepsilon < 1/2 \quad R_N < \varepsilon$ /2/

Из этого следует, что комбинаторный алгоритм лучше каждого данного алгоритма.

Когда исходные алгоритмы A_i имеют ошибочные вероятности ε_i соответственно, то ошибочная верояность комбинаторного алгоритма удовлетворяет неравенству

$$R_N \le \frac{1}{N} \cdot \left(\frac{1}{1 - 2 \max\limits_{1\le i\le N} \varepsilon_i}\right)^2 \quad \text{причем,} \qquad /3/$$

$$R_N \le \frac{4}{N^3} \cdot \left(\frac{\sum\limits_{i=1}^{N} \sqrt{\varepsilon_i(1-\varepsilon_i)}}{1-2\sum\limits_{i=1}^{N} \varepsilon_i/N}\right)^2 . \qquad /3.1/$$

1. ВВЕДЕНИЕ

Комбинаторный метод алгоритма распознавания образов используется для получения нового алгоритма распознавания с эффективностью большей чем эффективность каждого исходного алгоритма, или иначе говоря, получаемый алгоритм имеет вероятность меньше чем вероятность каждого данного алгоритма.

Этот метод основывается на правиле большинственных решений. Он формулируется следующим образом [1]:

Пусть даны N алгоритмов распознавания образов $A_1,\ldots,A_N$, классифицирующие объекты $\{s_i \mid i = \overline{1, n}\}$ на два класса K_1, K_2. Определим матрицу $\| a_{ij}^k \|_{n\times 2}$, $k = \overline{1,N}$ как следующее:

$$a_{ij}^k = \begin{cases} 1 \text{ если алгоритм } A_k \text{ относит объект } s_i \text{ к классу } K_j \\ 0 \text{ в обратном случае} \end{cases}$$

Тогда, комбинаторный алгоритм распознания образов, основывающийся на правиле большинственных решений, определит

$$D_j(s_i) = \begin{cases} 1 \text{ если } \sum\limits_{k=1}^{N} a_{ij}^k > \lfloor N/2 \rfloor \\ 0 \text{ в обратном случае} \end{cases}$$

где $D_j(s_i) = 1$ значит, что комбинаторный алгоритм распознава-

ния образов относит s_i к классу K_j, $i = \overline{1,n}$, $j = \overline{1,2}$.

Когда алгоритм $A_1,\ldots,A_N$ независим и каждый из них имеет ошибочную вероятность $\varepsilon > 0$, то ошибочная вероятность комбинаторного алгоритма определяется формулой:

$$R_N = \sum_{k>\lfloor N/2 \rfloor} \binom{N}{k} \varepsilon^k (1-\varepsilon)^{N-k} \qquad /4/$$

В параграфе § 2 этой статьи будет доказано, что $\forall \varepsilon: 0<\varepsilon\leq 1/2, R_N<\varepsilon$ /это неравенство показывает, что плучаемый комбинаторный алгоритм лучше каждого исходного алгоритма/ и

$$R_N \leq \exp\left(-N.\ln\left(1/2\sqrt{\varepsilon(1-\varepsilon)}\right)\right)$$

/Благодаря этому неравенству будет получен следующие интересный результат:

Если каждый из N данных алгоритмов имеет ошибочную вероятность $0 < \varepsilon < 1/2$, что скорость схождения R_N к нулю имеет экспонентный порядок при $N \to \infty$ ./

В том случае, когда алгоритмы A_i независимы и имеют ошибочные вероятности ε_i соответственно, то ошибочная вероятность R_N может определяться следующим образом:

$$R_N = \sum_{(\lambda_1,\ldots,\lambda_N)\in TA} \prod_{k=1}^{N} \varepsilon_k^{\lambda_k} (1-\varepsilon_k)^{(1-\lambda_k)},$$

$$TA=\{(\lambda_1,\ldots,\lambda_N) \mid \forall k = \overline{1,N} \quad \lambda_k \in \{0,1\},$$

$$\sum_{k=1}^{N} \frac{\lambda_k}{\sqrt{\varepsilon_k(1-\varepsilon_k)}} > \frac{1}{2} \sum_{k=1}^{N} \frac{1}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\}$$

Более того, $R_N \le \frac{1}{N} \cdot \left(\frac{1}{1 - 2 \max\limits_{1 \le i \le N} \varepsilon_i} \right)^2$ и

$$R_N \le \frac{4}{N^3} \left(\frac{\sum\limits_{i=1}^{N} \sqrt{\varepsilon_i (1 - \varepsilon_i)}}{1 - 2 \sum\limits_{i=1}^{N} \varepsilon_i / N} \right)^2 .$$

Это утверждение будет доказано в параграфе § 3.

2. ОЦЕНКА ОШИБОЧНОЙ ВЕРОЯТНОСТИ КОМБИНАТОРНОГО АЛГОРИТМА ДЛЯ НЕЗАВИСИМЫХ АЛГОРИТМОВ С РАВНЫМИ ОШИБОЧНЫМИ ВЕРОЯТНОСТЯМИ

Теорема 1.

$R_N = \sum\limits_{k > \lfloor N/2 \rfloor} \binom{N}{k} \varepsilon^k (1 - \varepsilon)^{N-k}$ и $R_N < \varepsilon$ для любого ε $0 < \varepsilon \le \frac{1}{2}$.

Доказательство.

Прежде всего, мы видим, что функция $f(\varepsilon) = \varepsilon^a (1-\varepsilon)^b$, $a \ge b > 0$, $0 < \varepsilon \le 1/2$ строгомонотонная, т.е. $\forall \varepsilon_1 \; \forall \varepsilon_2 \quad 0 < \varepsilon_1 < \varepsilon_2 \le \frac{1}{2}$

$f(\varepsilon_1) < f(\varepsilon_2)$. /5/

Поэтому требуемое неравенство эквивалентно

$$T = \sum_{k > \lfloor N/2 \rfloor} \binom{N}{k} \varepsilon^{k-1} (1-\varepsilon)^{N-k} < 1 \qquad /6/$$

Из-за того, что $k > \lfloor N/2 \rfloor$, то $k - 1 \ge N-k$, в результате неравенства /5/ мы получим

$$\varepsilon^{k-1} (1-\varepsilon)^{N-k} < \left(\frac{1}{2}\right)^{k-1} \left(1 - \frac{1}{2}\right)^{N-k} = \frac{1}{2^{N-1}} .$$

Поэтому

$$T < \sum_{k > \lfloor N/2 \rfloor} \binom{N}{k} \cdot \frac{1}{2^{N-1}} = \frac{1}{2^{N-1}} \cdot \sum_{k > \lfloor N/2 \rfloor} \binom{N}{k} .$$

При $N = 2m$

$$\sum_{k>\lfloor N/2\rfloor} \binom{N}{k} = \sum_{k=m+1}^{2m} \binom{2m}{k} = \frac{1}{2}\left(\sum_{k=0}^{m-1} \binom{2m}{k} + \sum_{k=m+1}^{2m} \binom{2m}{k}\right).$$

Из этого следует, что $T < \frac{1}{2^{2m}}\left(2^{2m} - \binom{2m}{m}\right) = 1 - \frac{\binom{2m}{m}}{2^{2m}} < 1.$

При $N = 2m + 1$

$$\sum_{k>\lfloor N/2\rfloor} \binom{2m+1}{k} = \sum_{k=m+1}^{2m+1} \binom{2m+1}{k} = \frac{1}{2} \sum_{k=0}^{2m+1} \binom{2m+1}{k} = 2^{2m} .$$

Отсюда следует, что $T < \frac{1}{2^{2m}} \cdot 2^{2m} = 1.$

При $\varepsilon > 0$, последовательность $\{R_N\}$ обладает свойством:

Теорема 2.

$$\forall \varepsilon: 0 < \varepsilon < 1/2,\ R_N = \sum_{k>\lfloor N/2\rfloor} \binom{N}{k} \varepsilon^k (1-\varepsilon)^{N-k} < \exp(-N.c(\varepsilon)),$$

где $c(\varepsilon) = \ln\left(\frac{1}{2\sqrt{\varepsilon(1-\varepsilon)}}\right) > 0$.

Доказательство.

Прежде всего, отмечаем, что при $0 < \varepsilon < 1/2$

$\ln((1-\varepsilon)/\varepsilon) > 0$ и $\ln(1/(2.\sqrt{\varepsilon(1-\varepsilon)})) > 0$.

При $\lambda > 0$ $\exp(\lambda.k) > \exp(\lambda.N/2)$ при $k > \lfloor N/2\rfloor$.

Отсюда следует, что $\exp(-\lambda.N/2) . \exp(\lambda.k) > 1$ /7/

Из-за того, что

$R_N = \sum_{k>\lfloor N/2\rfloor} \binom{N}{k} \varepsilon^k (1-\varepsilon)^{N-k}$ и /7/ получим

$$R_N < \exp(-\lambda.N/2) \sum_{k>\lfloor N/2 \rfloor} \binom{N}{k} \exp(\lambda.k) \cdot \varepsilon^k (1-\varepsilon)^{N-k}$$

$$< \exp(-\lambda.N/2) \cdot \sum_{k=0}^{N} \binom{N}{k} \exp(\lambda.k) \cdot \varepsilon^k \cdot (1-\varepsilon)^{N-k}$$

$$= \exp(-\lambda.N/2)\ (1-\varepsilon+\exp(\lambda).\varepsilon)^N = \left(\frac{1-\varepsilon}{\exp(\lambda/2)} + \varepsilon.\exp(\lambda/2)\right)^N .$$

Положив $\lambda = \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) > 0$, мы получим $\frac{1-\varepsilon}{\exp(\lambda/2)} = \varepsilon.\exp(\lambda/2) = \sqrt{\varepsilon(1-\varepsilon)}$. Поэтому

$$R_N < \left(2\sqrt{\varepsilon(1-\varepsilon)}\right)^N .$$

Положив $c(\varepsilon) = \ln\left(\frac{1}{2\sqrt{\varepsilon(1-\varepsilon)}}\right)$, имеем $R_N < \exp(-N.c(\varepsilon))$.

/Из факта, что $c(\varepsilon)$ - постоянная следует, что $R_N \to 0$ при $N \to \infty$, R_N имеет скорость схождения экспонентного порядка./

3. ОЦЕНКА ОШИБОЧНОЙ ВЕРОЯНОСТИ КОМБИНАТОРНОГО АЛГОРИТМА ПРИ РАЗЛИЧНЫХ ОШИБОЧНЫХ ВЕРОЯТНОСТЯХ АЛГОРИТМОВ

Правило решений комбинаторного алгоритма определяется следующим образом:

$$D_j(s_i) = \begin{cases} 1 \text{ если } \sum_{k=1}^{N} \frac{a_{ij}^k}{\sqrt{\varepsilon_k(1-\varepsilon_k)}} > \frac{1}{2} \sum_{k=1}^{N} \frac{1}{\sqrt{\varepsilon_k(1-\varepsilon_k)}} \\ 0 \text{ в обратном случае} \end{cases}$$

Теорема 3.

Если исходные алгоритмы A_i независимы и имеют ошибочные вероятности ε_i, соответственно, то ошибочная вероятность R_N может определяться следующим образом:

$$R_N = \sum_{(\lambda_1,\ldots,\lambda_N)\in TA} \prod_{k=1}^{N} \varepsilon_k^{\lambda_k}(1-\varepsilon_k)^{(1-\lambda_k)},$$

где $TA = \{(\lambda_1,\ldots,\lambda_N)/\forall k=\overline{1,N}\ \ \lambda_k \in \{0,1\},$

$$\sum_{k=1}^{N} \frac{\lambda_k}{\sqrt{\varepsilon_k(1-\varepsilon_k)}} > \frac{1}{2}\sum_{k=1}^{N} \frac{1}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\ \}.$$

Теорема 4.

При условии как в теореме 3 мы получим

$$R_N \le \frac{1}{N}\cdot\frac{1}{(1-2\max\limits_{1\le i\le N}\varepsilon_i)^2}\ .$$

Доказательство.

Из теоремы 3 следует, что

$$R_N \le \Big(\frac{1}{2}\cdot\sum_{k=1}^{N}\frac{(1-2\varepsilon_k)}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\Big)^{-2} \sum_{(\lambda_1,\ldots,\lambda_N)\in TA} \prod_{k=1}^{N} \varepsilon_k^{\lambda_k}\cdot(1-\varepsilon_k)^{(1-\lambda_k)}$$

$$\Big(\sum_{k=1}^{N}\frac{(\lambda_k-\varepsilon_k)}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\Big)^2.$$

Нетрудно видеть, что

$$\sum_{(\lambda_1,\ldots,\lambda_N)\in TA} \prod_{k=1}^{N} \varepsilon_k^{\lambda_k}(1-\varepsilon_k)^{(1-\lambda_k)}\Big(\sum_{k=1}^{N}\frac{(\lambda_k-\varepsilon_k)}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\Big)^2 \le N.$$

Поэтому

$$R_N \le \Big(\frac{1}{2}\sum_{k=1}^{N}\frac{(1-2\varepsilon_k)}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\Big)^{-2}\cdot N \quad /8/ \quad \le \Big(\frac{1}{2}\sum_{k=1}^{N}2(1-2\varepsilon_k)\Big)^{-2}\cdot N$$

$$\le [N(1-2\max_{1\le i\le N}\varepsilon_i)]^{-2}\cdot N = \frac{1}{N}\cdot\frac{1}{(1-2\max\limits_{1\le i\le N}\varepsilon_i)^2}\ .$$

Теорема 5.

$$R_N \le \frac{4}{N^3} \cdot \frac{\left(\sum_{k=1}^{N} \sqrt{\varepsilon_k(1-\varepsilon_k)}\right)^2}{\left(1 - 2\cdot\frac{1}{N}\cdot \sum_{k=1}^{N} \varepsilon_k\right)^2}$$

Доказательство.

Прежде всего мы докажем следующий результат:

Лемма /Неравенство Чебышева/:

Если последовательности $\{a_i\}$, $\{b_i\}$ удовлетворяют следущим условиям:

$\forall i > j \quad a_i \le a_j$, $b_i \ge b_j$, то

$$\left(\frac{1}{N}\cdot\sum_{i=1}^{N} a_i\right)\left(\frac{1}{N}\cdot\sum_{i=1}^{N} b_i\right) \ge \frac{1}{N}\cdot\sum_{i=1}^{N} a_i b_i \ .$$

Доказательство

Нетрудно видеть, что $\forall\, i, j \ (a_i - a_j)(b_i - b_j) \le 0$. Поэтому

$$\sum_{i,j} (a_i - a_j)(b_i - b_j) \le 0 \quad \text{или}$$

$$\sum_{i,j} (a_i b_i - a_j b_i - a_i b_j + a_j b_j) \le 0,$$

$$2N \cdot \sum_{i} a_i b_i - 2 \sum_{i,j} a_i b_j \le 0, \ \text{или}$$

$$N \sum_{i=1}^{N} a_i b_i \le \left(\sum_{i=1}^{N} a_i\right)\left(\sum_{j=1}^{N} b_j\right).$$

Отсюда мы можем получить неравенство

$$\left(\frac{1}{N}\sum_{i=1}^{N} a_i\right)\left(\frac{1}{N}\sum_{i=1}^{N} b_i\right) \ge \frac{1}{N}\sum_{j=1}^{N} a_i b_i \ .$$

Сейчас мы докажем теорему:

Из неравенства /8/ следует, что

$$R_N \le \left(\frac{1}{2}\sum_{k=1}^{N}\frac{(1-2\cdot\varepsilon_k)}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\right)^{-2}\cdot N \ .$$

Используя лемму может получиться неравенство

$$\left(\sum_{k=1}^{N}\frac{(1-2\cdot\varepsilon_k)}{\sqrt{\varepsilon_k(1-\varepsilon_k)}}\right)\left(\sum_{k=1}^{N}\sqrt{\varepsilon_k(1-\varepsilon_k)}\right) \ge N\cdot\left(\sum_{k=1}^{N}(1-2\varepsilon_k)\right) =$$

$$= N^2\left(1-\frac{2}{N}\sum_{k=1}^{N}\varepsilon_k\right) \ .$$

$$R_N \le \frac{4N\left(\sum_{k=1}^{N}\sqrt{\varepsilon_k(1-\varepsilon_k)}\right)^2}{N^4\left(1-\frac{2}{N}\sum_{k=1}^{N}\varepsilon_k\right)^2} = \frac{4}{N^3}\cdot\frac{\left(\sum_{k=1}^{N}\sqrt{\varepsilon_k(1-\varepsilon_k)}\right)^2}{\left(1-2\sum_{k=1}^{N}\varepsilon_k/N\right)^2} \ .$$

Выражаем большую благодарность профессору Бак Хынг Кхангу за постоянную помощь во время работы.

Мы также благодарны нашей коллеге Нгуен Тхать Тзу - из Ханойского политехнического института за хорошие советы и всем нашим коллегам в лаборатории за их симпатию и поддержку нашей работы.

Л И Т Е Р А Т У Р А

[1] Hoang Kiém: Some methods improving the efficiency of pattern recognition algorithms. Computer and artificial intelligence, 3/1984/, N° 4, pp. 347-359.

[2] Gaillat G.: Borne superieure de la probabilité d'eurreur avec la régla de decision majoritaire. Congres AFCET - IRIA, 1978, pp. 230-236.

Alakfelismerési kombinatorikus algoritmusok hatékonyságának becslése

Ngo Kuoc Tao, Hoang Kiem

Összefoglaló

Legyenek $A_1, A_2, \ldots, A_N$ alak-felismerési algoritmusok, amelyek n objektumot két kategóriába sorolnak be, és tegyük fel, hogy /egymástól függetlenül/ ε_i valószínüséggel hibáznak az A_i-k. Az A_i-k segitségével a szerzők egy u.n. "alakfelismerési kombinatorikus algoritmust" definiálnak és annak R_N "hibázási" valószínüségét. A szerzők bebizonyitják, hogy ha $\varepsilon_1 = \varepsilon_2 = \ldots = \varepsilon_N = \varepsilon$ és $0 < \varepsilon \leq \frac{1}{2}$, akkor $R_N < \varepsilon$ és $R_N \leq \exp(-N \ln(1/(2\sqrt{\varepsilon(1-\varepsilon)})))$, az általános esetben pedig $R_N \leq N^{-1}(1-2 \max_{1 \leq i \leq N} \varepsilon_i)^{-2}$ és $R_N \leq 4N^{-3} (\Sigma\sqrt{\varepsilon_i(1-\varepsilon_i)})^2 (1-2\Sigma \varepsilon_i/N)^{-2}$.

Estimation for efficiency of pattern-recognition combinatorial algorithms

Ngoc Kuoc Tao, Hoang Kiem

Summary

Let $A_1, A_2, \ldots, A_N$ be pattern-recognition algorithms which classify n objects into two categories. Assume that, independently of each other, the probability that A_i classifies wrongly is ε_i. Using A_i, the authors define a "pattern recognition combinatorial algorithm" and its probability R_N of wrong classification. They prove that if $\varepsilon_1 = \varepsilon_2 = \ldots = \varepsilon_N = \varepsilon$ and $0 < \varepsilon \leq 1/2$ then $R_N < \varepsilon$ and $R_N \leqslant \exp(-N \ln(1/(2\sqrt{\varepsilon(1-\varepsilon)}\,)))$, and in the general situation $R_N \leq N^{-1}(1-2 \max_{1\leq i\leq N} \varepsilon_i)^{-2}$ and $R_N \leq 4N^{-3}(\Sigma \sqrt{\varepsilon_i(1-\varepsilon_i)}\,)^2(1 - 2\,\Sigma\, \varepsilon_i/N)^{-2}$.

MTA SZTAKI Közlemények 35/1986 p. 147-168

ON THE SEPARABLE AND ANNULING SETS OF VARIABLES FOR THE FUNCTIONS

Sl. SHTRAKOV

Higher Pedagogical Institute
Blagoevgrad, Bulgaria

1. INTRODUCTION

In the past 20-30 years the investigations of the discrete functions with respect to the separability or inseparability of the sets of variables are intensified. These properties of the functions and sets arise naturally in the consideration of some questions in synthesis schemes and functional dependences in finite algebras of discrete functions. The paper is a continuation of the studies of the separable and inseparable sets of variables which are begun by K.N.Čimev and the author.

In Section 2 we study the annuling sets of variables for the functions.There are some assertions which describe these sets.

In Section 3 we introduce s-system and c-system of an arbitrary family of non-empty sets. This section has a set-theoritical nature and results obtained here are used in the next section for discussions and generalizations of two very important theorems proved by K.N.Čimev. The connection of annuling, separable and inseparable sets of variables is considered in Section 4.

2. ANNULING SETS OF VARIABLES FOR THE FUNCTIONS

We use some notations and terminology from [1,2,3]

For A any nonvoid set, A^n denotes the set of all n-tuples of elements in A. For n $(n>0)$ an n-ary function on A is a function $f: A^n \to A$.

A function $f(x_1,x_2,\dots,x_n)$ on A is said to depend on its i-th variable x_i if there exist constants

$$a,b,c_1,c_2,\dots,c_{i-1},c_{i+1},\dots,c_n \in A$$

for which

$$f(c_1,c_2,\dots,c_{i-1},a,c_{i+1},\dots,c_n) \neq$$
$$f(c_1,c_2,\dots,c_{i-1},b,c_{i+1},\dots,c_n).$$

If the function f depends on x_i then x_i is called essential variable for f. The set of all essential variables for f is denoted by R_f [1,2]

For $M = \{x_{i_1},x_{i_2},\dots,x_{i_m}\}$ any arbitrary set of essential variables for f and for any m-tuple $(c_{i_1},c_{i_2},\dots,c_{i_m}) \in A^m$ we may obtain from f the following function on A

$$g = f(x_{i_1}=c_{i_1},x_{i_2}=c_{i_2},\dots,x_{i_m}=c_{i_m})$$

which is called a subfunction of f. When g is a subfunction of f we will write $g \prec f$.

A set M of essential variables for the function f is called separable for f if there is an r-tuple $(c_{j_1},c_{j_2},\dots,c_{j_r}) \in A^r$ of values for the variables in the set $R_f \setminus M = \{x_{j_1},x_{j_2},\dots,x_{j_r}\}$ such that

$$M \subseteq R_{f(x_{j_1}=c_{j_1},x_{j_2}=c_{j_2},\dots,x_{j_r}=c_{j_r})}.$$

The set of all separable sets for f is denoted by S_f [1,2].

A set M ($M \subseteq R_f$) is called c-separable for f with respect to $N = \{x_{j_1}, x_{j_2}, \ldots, x_{j_s}\}$ ($N \subseteq R_f$) if for each s-tuple $(c_{j_1}, c_{j_2}, \ldots, c_{j_s}) \in A^s$ of the variables in N it is hold

$$M \subseteq R_f(x_{j_1}=c_{j_1}, x_{j_2}=c_{j_2}, \ldots, x_{j_s}=c_{j_s}).$$

The set of all c-separable sets for f with respect to N will be denoted by $S^*_{f,N}$.

Some of the properties of the c-separable sets are discussed in [3,4].

When the set M isn't separable for f then it is called inseparable for f. So, M is inseparable for f if for each r-tuple $(c_{j_1}, c_{j_2}, \ldots, c_{j_r}) \in A^r$ of values for the variables in $R_f \setminus M$ it is held

$$M \nsubseteq R_f(x_{j_1}=c_{j_1}, x_{j_2}=c_{j_2}, \ldots, x_{j_r}=c_{j_r}).$$

DEFINITION 2.1. A set $N = \{x_{i_1}, x_{i_2}, \ldots, x_{i_s}\}$ of essential variables for the function f is called annuling of M ($M \subseteq R_f$) if for each s-tuple $(d_{i_1}, d_{i_2}, \ldots, d_{i_s}) \in A^s$ of values for the variables in N it is held

$$M \nsubseteq R_f(x_{i_1}=d_{i_1}, x_{i_2}=d_{i_2}, \ldots, x_{i_s}=d_{i_s})$$

and N is minimal with respect to this property i.e. if $N_1 = \{x_{k_1}, x_{k_2}, \ldots, x_{k_t}\} \subseteq N$ and for each t-tuple $(d_{k_1}, d_{k_2}, \ldots, d_{k_t}) \in A^t$ of values for the variables in N_1 it is held

$$M \nsubseteq R_f(x_{k_1}=d_{k_1}, x_{k_2}=d_{k_2}, \ldots, x_{k_t}=d_{k_t})$$

then $N_1 = N$.

Denote by $\overset{o}{An}(M,f)$ the set of all annuling sets of M for f and by $\overset{o}{Uan}(M,f)$ the union of the sets belonging to $An^c(M,f)$ i.e.

$$Uan^o(M,f) = \bigcup_{N \in An^o(M,f)} N$$

It is easy to see that each non-empty subset M_1 of M for which $|M_1| = 1$ is an annuling set of M for f.

Besides, if $N \in An^o(M,f)$ and $M \cap N \neq \emptyset$ then obviously $N \subseteq M$ and $|N| = 1$. This note gives us a possibility to define as trivial annuling sets of M, these sets N for which $N = \{x_{J_k}\}$, and $x_{j_k} \in M$. Thus we will consider only non-trivial annuling sets of M for f i.e. these sets N for which $N \in An^o(M,f)$ and $N \cap M = \emptyset$. Denote by An(M,f) the set of all non-trivial annuling sets of M for f and by Uan(M,f) the union of all sets belonging to An(M,f).

LEMMA 2.1. If M is an inseparable non-empty set of essential variables for the function f then M has at least one non-trivial annuling set for f .

LEMMA 2.2. Let $M \subseteq R_f$ and $N = \{x_{j_1}, x_{j_2}, \ldots, x_{j_s}\} \subseteq R_f$ .If for each s-tuple $(d_{j_1}, d_{j_2}, \ldots, d_{j_s}) \in A^s$ of values for the variables in N

$$M \nsubseteq R_f(x_{j_1}=d_{j_1}, x_{j_2}=d_{j_2}, \ldots, x_{j_s}=d_{j_s})$$

holds then there exists at least one subset N_1 of N such that

$$N_1 \in An(M,f).$$

LEMMA 2.3. If $N \in An(M,f)$ then $N \in An(R,f)$ where R is an arbitrary set of essential variables for f such that $M \subseteq R$ and $N \cap R = \emptyset$.

LEMMA 2.4. If $N \in An(M,f)$ then for every two sets N_1 $(N_1 \subseteq N)$ and M_1 $(M_1 \subseteq M)$ it is hold

$$N_1 \notin An(M_1,f).$$

PROOF. If $M_1 = M$ then the lemma is trivial.
Let $M_1 \subsetneq M$. Suppose that the lemma is false and let $N_1 \in An(M_1,f)$. By Lemma 2.3 it follows $N_1 \in An(R,f)$ where R is a set for which $R \cap N_1 = \emptyset$ and $M_1 \subseteq R$. Obviously $M_1 \subseteq M$ and $M \cap N_1 = \emptyset$, since $M \cap N = \emptyset$.

So,
$$N_1 \in An(M,f)$$
and
$$N_1 \subsetneq N.$$
A contradiction. The lemma is proved.

THEOREM 2.5. Let $M \in An(L,f)$ and $x_{i_p} \in M$. If R is a set belonging to $An(L \cup \{x_{i_p}\},f)$ then there exists at least one subset R_1 of $R \cup M \setminus \{x_{i_p}\}$ such that $R_1 \in An(L,f)$.

PROOF. Without loss of generality, assume that
$M = \{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\}$, $N = \{x_{k_1}, x_{k_2}, \ldots, x_{k_l}\}$ and
$$R = \{x_{j_1}, x_{j_2}, \ldots, x_{j_r}\}.$$
Let $c_{j_1}, c_{j_2}, \ldots, c_{j_r}, c_{i_1}, c_{i_2}, \ldots, c_{i_{p-1}}, c_{i_{p+1}}, \ldots, c_{i_m}$ be

arbitrary values of the variables in $R \cup M \setminus \{x_{i_p}\}$

At first, we shall prove that

(1) $\quad L \cap R_{f_1} \neq L$

where

$$f_1 = f(x_{i_1}=c_{i_1}, x_{i_2}=c_{i_2}, \ldots, x_{i_{p-1}}=c_{i_{p-1}}, x_{i_{p+1}}=c_{i_{p+1}}, \ldots,$$

$$x_{i_m}=c_{i_m}, x_{j_1}=c_{j_1}, x_{j_2}=c_{j_2}, \ldots, x_{j_r}=c_{j_r})$$

Suppose that (1) isn't held i.e. $L \subseteq R_{f_1}$.

By $R \in An(L \cup \{x_{i_p}\}, f)$ it follows that $x_{i_p} \notin R_{f_1}$. Consequently

$$f_1 = f_1(x_{i_p}=d_{i_p})$$

for each constant $d_{i_p} \in A$. Now, $L \subseteq R_{f_1}$ implies that

$$L \subseteq R_{f_2}$$

where

$$f_2 = f(x_{i_1}=c_{i_1}, x_{i_2}=c_{i_2}, \ldots, x_{i_{p-1}}=c_{i_{p-1}}, x_{i_p}=d_{i_p}, x_{i_{p+1}}=c_{i_{p+1}}, \ldots,$$

$$x_{i_m}=c_{i_m}).$$

Hence $M \notin An(L,f)$. A contradiction. Consequently (1) always is held. By Lemma 2.2 it follows that there exists at least one subset R_1 of $R \cup M \setminus \{x_{i_p}\}$ for which $R_1 \in An(L,f)$ and $R \neq \emptyset$.

The proof is complete.

LEMMA 2.6. If M is an inseparable non-empty set of essential variables for the function f then $|M| \geq 2$.

PROOF. Suppose that the lemma is false i.e. $M = \{x_i\}$. Now, $M \notin S_f$ implies $x_i \notin R_f$ and $M \cap R_f = \emptyset$. A contradiction. Consequently $|M| \geq 2$.

3. s - SYSTEMS AND C - SYSTEMS FOR A FAMILY OF NON-EMPTY SETS

Let $\Omega = \{P_1, P_2, \ldots, P_s\}$ be a family of non-empty sets.

DEFINITION 3.1. A system $\Sigma = \{x_1, x_2, \ldots, x_t\}$ is called s - system of Ω if it is held

(i) for each $P_i \in \Omega$ there exists $x_{i_j} \in \Sigma$ such that $x_{i_j} \in P_i$ and

(ii) for each $x_k \in \Sigma$ there exists $P_{k_1} \in \Omega$ such that $P_{k_1} \cap \{x_1, x_2, \ldots, x_t\} = \{x_k\}$.

DEFINITION 3.2. A system $\Sigma = \{x_1, x_2, \ldots, x_t\}$ is called c - system of Ω if Σ is an s-system of Ω and for each other s-system $\Sigma' = \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$ of Ω it is held $t \leq p$.

LEMMA 3.1. For any family $\Omega = \{P_1, P_2, \ldots, P_s\}$ of non-empty sets there exists at least one s-system of Ω.

PROOF. We will use induction on s, the number of the sets belonging to Ω.

Obviously, all families Ω of one set P_1 has at least

one s-system, since $P_1 \neq \emptyset$ and $\sum = \{x_i\}$ is an s-system of Ω for each $x_i \in P_1$.

Assume the lemma is true for $s \leq k$ and suppose $s = k+1$. So, $\Omega = \{P_1, P_2, \ldots, P_k, P_{k+1}\}$, $P_i \neq \emptyset$ for $i=1,2,\ldots,k+1$.

At first, note that there exists a set $P_{j_o} \in \Omega$ such that $P_i \in \Omega$, $i \neq j_o$ imply $P_i \setminus P_{j_o} \neq \emptyset$. The existence of the set P_{j_o} follows by $P_i \neq \emptyset$ for $i=1,2,\ldots,k+1$. Let x_o be an arbitrary element of P_{j_o}.

If for each $i=1,2,\ldots,k+1$, x_o belongs to P_i then obviously $\sum = \{x_o\}$ is an s-system of Ω and lemma is proved in this case.

Now, let $\Delta = \{P_1, P_2, \ldots, P_m\}$ be family of all sets of Ω which do not contain x_o i.e. $x_o \notin P_i$ for $i=1,2,\ldots,m$. Obviously $m < k+1$. Denote by Ω' the family obtained from these sets as follows

$$\Omega' = \{P_1', P_2', \ldots, P_m'\}$$

where

$$P_i' = P_i \setminus P_{j_o} \qquad \text{for } i=1,2,\ldots,m.$$

By the induction assumption we conclude that there exists at least one s-system $\sum' = \{x_1', x_2', \ldots, x_p'\}$ of Ω'. We will next show that

$$\sum = \{x_o, x_1', x_2', \ldots, x_p'\}$$

is an s-system of Ω.

At first, prove that the condition (i) of Definition 3.1

is satisfied.

Let $P_i \in \Omega$. If $x_o \in P_i$ then (i) is satisfied. If $x_o \notin P_i$ then $P_i \setminus P_{j_o} \in \Omega'$ end there is an element $x_1' \in \Sigma'$ which belongs to $P_i \setminus P_{j_o}$. Consequently $x_1' \in P_i$.

So, (i) is satisfied in the general case.

Now, prove that (ii) is satisfied, too.

Let x_1 be an arbitrary element from Σ. If $x_1 = x_o$ then it is clear that $x_1 \in P_{j_o}$. Note that for each $P_i' \in \Omega'$ it is held

$$P_i' \cap P_{j_o} = \emptyset.$$

Besides, Σ' is an s-system of Ω' and it implies

$$P_{j_o} \cap \{x_1', x_2', \ldots, x_p'\} = \emptyset.$$

Consequently

$$P_{j_o} \cap \{x_o, x_1', x_2', \ldots, x_p'\} = \{x_o\}.$$

So, (ii) is satisfied in this case.

Let $x_1 = x_i'$ for some i belonging to $\{1, 2, \ldots, p\}$. Let P_r' be a set of Ω' such that

$$(2) \qquad P_r' \cap \{x_1', x_2', \ldots, x_p'\} = \{x_i'\}.$$

By $P_r' \in \Omega'$ it follows that there exists a set $P_t \in \Delta$ such that $P_r' = P_t \setminus P_{j_o}$. We will prove that

$$P_t \cap \{x_o, x_1', x_2', \ldots, x_p'\} = \{x_i'\}$$

Assume that

$$P_t \cap \{x_o, x_1', x_2', \ldots, x_p'\} = R$$

where $R \subseteq \Sigma$ and obviously $x_i' \in R$.

Suppose that $x_o \in R$. Hence $x_o \in P_t$ and $P_t \notin \Delta$. A contradiction. Consequently $x_o \notin R$. Since $P'_r = P_t \setminus P_{j_o}$ and by (2) it follows that

$$P_t \setminus P_{j_o} \cap \{x_o, x'_1, x'_2, \ldots, x'_p\} = \{x'_i\}.$$

Consequently $R \setminus \{x'_i\} \subseteq P_{j_o}$. Besides

$$P_{j_o} \cap \{x_o, x'_1, x'_2, \ldots, x'_p\} = \{x_o\}$$

and it follows

$$R \setminus \{x'_i\} \subseteq \{x_o\}.$$

Now, by $x_o \notin R$ it follows that $R = \{x'_i\}$. The proof is complete.

DEFINITION 3.3. A family $\Omega = \{P_1, P_2, \ldots, P_s\}$ of non-empty sets is called Sperner - family if $P_i, P_j \in \Omega$ and $P_i \neq P_j$ imply $P_i \not\subseteq P_j$.

THEOREM 3.2. Let $\Omega = \{P_1, P_2, \ldots, P_s\}$ be a Sperner - family of non-empty sets. For each $P_i \in \Omega$ and for each $x_{j_i} \in P_i$ there exists at least one s-system of Ω which contains x_{j_i}.

PROOF. Let P_i be an arbitrary set of Ω and x_{j_i} be an arbitrary element from P_i. Since Ω is a Sperner - family it follows that for each $k=1,2,\ldots,s$, $k \neq i$

$$P_k \setminus P_i \neq \emptyset$$

holds.

Denote by $\Delta = \{P_1, P_2, \ldots, P_m\}$ the family of all sets of Ω for which $P_k \setminus P_i \neq \emptyset$ for $k=1,2,\ldots,m$. It is clear that $m = s-1$, and $\Omega \setminus \Delta = \{P_i\}$. Let $P'_k = P_k \setminus P_i$ for each $k=1,2,\ldots,m$. So, we obtain a new family $\Omega' = \{P'_1, P'_2, \ldots, P'_m\}$ of non-empty sets. By Lemma 3.1 there is at least one s-system $\Sigma' = \{x'_1, x'_2, \ldots, x'_t\}$ of Ω'.

By similar arguments as in proof of Lemma 3.1 it may be proved that

$$\Sigma = \{x_{j_i}, x'_1, x'_2, \ldots, x'_t\}$$

is an s-system of Ω. The proof of the theorem is complete.

This theorem is a sufficient condition for this : For each x_i belonging to some $P_i \in \Omega$ there exists at least one s-system of Ω containing x_i. But it isn't a necessary condition for this.

Example 3.1. Let $P_1 = \{1,2\}$, $P_2 = \{1,3\}$, $P_3 = \{2,4\}$, $P_4 = \{3\}$. Obviously, all s-systems of $\Omega = \{P_1, P_2, P_3, P_4\}$ are : $\Sigma_1 = \{1,4\}$ and $\Sigma_2 = \{2,3\}$. It is easy to see that Ω isn't a Sperner - family , but for each $x_i = 2,1,3,4$ there is at least one s-system of Ω containing x_i.

Example 3.2. Let $P_1 = \{1,2,3\}$, $P_2 = \{2\}$ and $P_3 = \{3\}$. It is easy to see that unique s-system of $\Omega = \{P_1, P_2, P_3\}$ is $\Sigma = \{2,3\}$. So , $1 \in P_1$ but there doesn't exist an s-system of Ω which contains 1.

THEOREM 3.3. Let $\Omega = \{P_1, P_2, \ldots, P_s\}$ be an arbitrary family

of non-empty sets. Let $x_i \in \bigcup_{P_k \in \Omega} P_k$. There doesn't exist an s-system of Ω which contains x_i if and only if

$$(\forall P_k \in \Omega)\ x_i \in P_k \Rightarrow (\exists P_r \in \Omega)\ P_r \subset P_k \ \&\ x_i \notin P_r.$$

PROOF. Let $P_{i_1}, P_{i_2}, \ldots, P_{i_m}$ be all sets of Ω for which $x_i \in P_{i_k}$ for $k=1,2,\ldots,m$ and $P_{t_1}, P_{t_2}, \ldots, P_{t_m}$ be sets of Ω such that

$$P_{t_l} \subset P_{i_l} \ \&\ x_i \notin P_{t_l} \quad \text{for } l=1,2,\ldots,m.$$

Suppose that there exists an s-system $\Sigma = \{x_1, x_2, \ldots, x_p\}$ of Ω which contains x_i. By (ii) of Definition 3.1 it follows that there exists a set $P_{i_j} \in \Omega$ ($1 \leq j \leq m$) such that

$$(3)\quad P_{i_j} \cap \{x_1, x_2, \ldots, x_p\} = \{x_i\}.$$

Now, $P_{t_j} \subset P_{i_j}$, $x_i \notin P_{t_j}$ and (i) of Definition 3.1 imply that there exists a distinct of x_i element $x_n \in \Sigma$ such that $x_n \in P_{t_j}$. Consequently $x_n \in P_{i_j}$ i.e. (3) isn't true. A contradiction. Hence, there doesn't exist an s-system of Ω which contains x_i.

Let us assume that there exists a set $P_{i_1} \in \Omega$ such that P_{i_1} contains x_i and

$$(\forall P_r \in \Omega)\quad P_r \not\subseteq P_{i_1} \quad \text{or} \quad x_i \in P_r.$$

Obviously, we may choose a set $P_{j_o} \subseteq P_{i_1}$ for which $x_i \in P_{j_o}$ and

$$(\forall P_r \in \Omega)\ P_r \not\subseteq P_{j_o}.$$

Now, by similar arguments as in proof of Lemma 3.1 it follows that there exists at least one s-system Σ of Ω for which $x_i \in \Sigma$. The proof is complete.

PROPOSITION 3.4. If M is an inseparable non-empty set of essential variables for the function f then the family An(M,f) is a Sperner - family.

This proposition follows immediately by Lemma 2.4.

COROLLARY. For each essential variable x_i for the function f ,belonging to Uan(M,f) there exists at least one s-system $\sum$ of An(M,f) which contains x_i, where $M \notin S_f$ and $M \neq \emptyset$.

4. SEPARABLE AND MAXIMAL SEPARABLE SETS OF VARIABLES FOR THE FUNCTIONS

The following two theorems are proved by K.Cimev [1,2] . The first of these theorems we provide with a new proof.

THEOREM 4.1. Let $f(x_1,x_2,\dots,x_n)$ be a function on A. If M is a proper separable for f subset of R_f then there is at least one variable x_t belonging to $R_f \setminus M$ such that

$$M \cup \{x_t\} \in S_f.$$

PROOF. Without loss of generality , assume that

$$R_f = \{x_1,x_2,\dots,x_n\} \quad \text{and} \quad M = \{x_1,x_2,\dots,x_m\} \ , \ m < n.$$

Let $(c_{m+1},c_{m+2},\dots,c_n) \in A^{n-m}$ be an n-m - tuple of values for the variables in $R_f \setminus M$ such that

$$M = R_{f_1}$$

where

$$f_1 = f(x_{m+1}=c_{m+1},x_{m+2}=c_{m+2},\dots,x_n=c_n).$$

For each p $(m \lessdot p \leqq n)$ denote by f_p the following function

$$f_p = f(x_{m+1}=c_{m+1}, x_{m+2}=c_{m+2}, \ldots, x_{p-1}=c_{p-1}, x_{p+1}=c_{p+1}, \ldots, x_n=c_n).$$

Suppose that the theorem is false. Consequently for each p $(m \lessdot p \leqq n)$ it is held $x_p \notin R_{f_p}$ i.e. $f_1 = f_p$. This implies that

$$f_{m+1}(x_{m+1}=c_{m+1}) = f(x_{m+2}=c_{m+2}, x_{m+3}=c_{m+3}, \ldots, x_n=c_n) = f_1$$

$$f_{m+2}(x_{m+2}=c_{m+2}) = f(x_{m+3}=c_{m+3}, x_{m+4}=c_{m+4}, \ldots, x_n=c_n) = f_1$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$f_n(x_n=c_n) = f = f_1.$$

Hence $R_f = R_{f_1}$ = M.A contradiction. Consequently there is a variable x_t belonging to $R_f \setminus M$ for which $M \cup \{x_t\}$ is separable for f. The theorem is proved.

COROLLARY 1. If $M \in S_f$, $|M| = m$ and $|R_f| = n$ $(m \lessdot n)$ then for each p $(0 \leqq p \leqq n-m)$ there exist at least p variables $x_{i_1}, x_{i_2}, \ldots, x_{i_p}$ belonging to $R_f \setminus M$ such that

$$M \cup \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\} \in S_f .$$

COROLLARY 2. If f_1 is a subfunction of f and $R_{f_1} \subsetneq R_f$ $(|R_{f_1}| = m, |R_f| = n)$ then there exist $n-m+1$ subfunctions $f_1, f_2, \ldots, f_{n-m+1}$ of f such that

$$f_1 \prec f_2 \prec \ldots \prec f_{n-m+1} = f$$

and

$$|R_{f_i}| = m+i-1 \quad \text{for} \quad i=1,2,\ldots,n-m+1.$$

DEFINITION 4.1. Let $R \subseteq R_f$ and $R \notin S_f$. A subset R_1 of R is called maximal separable of R for f if $R_1 \in S_f$ and

$$(\forall R_2 \subset R)\ R_2 \in S_f \ \&\ R_1 \subseteq R_2 \Rightarrow R_2 = R_1.$$

Denote by Max(R,f) the set of all maximal separable subsets of R.

THEOREM 4.2. Let $f(x_1,x_2,\ldots,x_n)$ be a function for which $\emptyset \neq R \subseteq R_f$ and $R \notin S_f$. If $R_1 \in \mathrm{Max}(R,f)$ then for every variables $x_{i_1},x_{i_2},\ldots,x_{i_m}$ belonging to $R \setminus R_1$ and for every their values $d_{i_1},d_{i_2},\ldots,d_{i_m}$ the following subfunction

$f_1 = f(x_{i_1}=d_{i_1},x_{i_2}=d_{i_2},\ldots,x_{i_m}=d_{i_m})$ depends at least on one variable belonging to $R_f \setminus R$ and $R_1 \in S_{f_1}$.

So, this theorem insists that $R_1 \in S^{*}_{f,R\setminus R_1}$ and $R_1 \in S_{f_1}$.

Now,we give some generalizations of the previous two theorems.

THEOREM 4.3. Let M be an inseparable non-empty set for the function f on A and $\mathrm{An}(M,f) = \{N_1,N_2,\ldots,N_s\}$. If $\Sigma = \{x_{j_1},x_{j_2},\ldots,x_{j_t}\}$ is an s-system of An(M,f) then

$$M \cup \Sigma \in S_f.$$

PROOF. By Lemma 2.1 it follows that $\mathrm{An}(M,f) \neq \emptyset$ and Lemma 2.6 implies that $|M| \geq 2$.

Assume that $R = R_f \setminus (M \cup \Sigma) = \{x_{k_1},x_{k_2},\ldots,x_{k_r}\}$. Since Σ is an s-system of An(M,f) and $N_1,N_2,\ldots,N_s$ are all annuling sets of M for f then by Lemma 2.4 it follows that

there exists an r-tuple $(c_{k_1}, c_{k_2}, \ldots, c_{k_r}) \in A^r$ of values for the variables in R such that $M \subseteqq R_{f_1}$ where

$$f_1 = f(x_{k_1}=c_{k_1}, x_{k_2}=c_{k_2}, \ldots, x_{k_r}=c_{k_r}).$$

We will prove that $\sum \subseteqq R_{f_1}$.

Suppose , this is false and without loss of generality assume that

$$x_{j_1} \notin R_{f_1}$$

i.e. $f_1 = f_1(x_{j_1}=d_{j_1})$ for each constant $d_{j_1} \in A$.

Let N_p be an annuling set of M for which

$$N_p \cap \sum = \{x_{j_1}\}.$$

The existence of this annuling set N_p follows by Lemma 3.1. Again, without loss of generality assume that $N_p = \{x_{k_1}, x_{k_2}, \ldots, x_{k_v}, x_{j_1}\}$, $v \leqq r$.

Since

$$R_{f_1} \cap \{x_{k_1}, x_{k_2}, \ldots, x_{k_v}\} = \emptyset$$

and $x_{j_1} \notin R_{f_1}$ it follows $N_p \cap R_{f_1} = \emptyset$.

Consequently for each v+1 - tuple $(d_{k_1}, d_{k_2}, \ldots, d_{k_v}, d_{j_1})$ belonging to A^{v+1} it is held

$$f_1 = f_1(x_{k_1}=d_{k_1}, x_{k_2}=d_{k_2}, \ldots, x_{k_v}=d_{k_v}, x_{j_1}=d_{j_1}).$$

By $M \subseteqq R_{f_1}$ it follows that $N_p \notin An(M,f)$. A contradiction. So, our supposition is false i.e. $x_{j_1} \in R_{f_1}$. In the same way as for x_{j_1} it may be proved that $x_{j_l} \in R_{f_1}$ for $l=1,2,\ldots,t$. Hence

$$M \cup \sum \subseteqq R_{f_1}.$$

Now, by $R_f \setminus R = M \cup \Sigma$ it follows that $M \cup \Sigma \in S_f$. The proof of the theorem is complete.

DEFINITION 4.2. Let $f(x_1,x_2,\ldots,x_n)$ be a function on A. If $M \subseteq R_f$ and Σ is an c-system of $An(M,f)$ when $M \notin S_f$ or $\Sigma = \emptyset$ when $M \in S_f$ then the number $C(M,f) = |\Sigma|$ is called degree of inseparability of M for f.

THEOREM 4.4. Let M be a set of essential variables for the function f. For each p ($C(M,f) \leq p \leq |R_f| - |M|$) there exist p variables $x_{j_1},x_{j_2},\ldots,x_{j_p}$ belonging to $R_f \setminus M$ such that

$$M \cup \{x_{j_1},x_{j_2},\ldots,x_{j_p}\} \in S_f.$$

PROOF. If $M \in S_f$ then $An(M,f) = \emptyset$ and the theorem follows in this case by Corollary 1 of Theorem 4.1.

If $M \notin S_f$ then $An(M,f) \neq \emptyset$ and let $\Sigma = \{x_{j_1},x_{j_2},\ldots, x_{j_t}\}$ be an c-system of $An(M,f)$. Now, by Theorem 4.3 it follows that

$$M \cup \{x_{j_1},x_{j_2},\ldots,x_{j_t}\} \in S_f.$$

By $C(M,f) = t$ it follows that $p-t \geq 0$ and by Corollary 1 of Theorem 4.1 we conclude that there exist p-t variables $x_{j_{t+1}},x_{j_{t+2}},\ldots,x_{j_p}$ belonging to $R_f \setminus (M \cup \{x_{j_1},x_{j_2},\ldots,x_{j_t}\})$ such that

$$M \cup \{x_{j_1},x_{j_2},\ldots,x_{j_p}\} \in S_f.$$

The proof is complete.

COROLLARY. If $M \subseteq R_f$ then there are $n-C(M,f)-m$ subfunctions of f , $f_1,f_2,\ldots,f_{n-C(M,f)-m}$ such that $M \subseteq R_{f_1}$,

$An(M,f) = \{\{x_2,x_3,x_4\}\}$ and $T \subset Uan(M,f)$ but $M \cup T \notin S_f$.

Let $M \subseteq R_f$ and $N = \{x_1, x_2, \ldots, x_s\} \subseteq R_f$. Denote by $D(M,N,f)$ the set of all variables $x_i \in N$ for which there is an s-tuple $(c_1,c_2,\ldots,c_s) \in A^s$ of values for the variables in N such that

$$M \in S_f(x_1=c_1, x_2=c_2, \ldots, x_{i-1}=c_{i-1}, x_{i+1}=c_{i+1}, \ldots, x_s=c_s)$$

and $$M \notin R_f(x_1=c_1, x_2=c_2, \ldots, x_{i-1}=c_{i-1}, x_i=c_i, x_{i+1}=c_{i+1}, \ldots, x_s=c_s)$$

Let $M \notin S_f$, $M \subseteq R_f$, $M_1 \subseteq M$ and $An(M,f) = \{N_1, N_2, \ldots, N_s\}$ Denote by $Ud(M_1,f)$ the following set

$$Ud(M_1,f) = \bigcup_{i=1}^{s} D(M_1,N_i,f).$$

THEOREM 4.6. Let M be an inseparable non-empty set of essential variables for the function f and $M_1 \in Max(M,f)$. If $M_2 = M \setminus M_1$ then

$$M_1 \cup Ud(M_1,f) \in S^*_{f,M_2}.$$

PROOF. Let $M_2 = \{x_{m+1}, x_{m+2}, \ldots, x_p\}$ and $(d_{m+1}, d_{m+2}, \ldots, d_p)$ by an arbitrary p-m - tuple of values for the variables in M_2. Let x_i be an arbitrary variable in $Ud(M_1,f)$. It is sufficient to prove that $M_1 \cup \{x_i\} \subseteq R_{f_1}$ where

$$f_1 = f(x_{m+1}=d_{m+1}, x_{m+2}=d_{m+2}, \ldots, x_p=d_p)$$

Theorem 4.2 implies that $M_1 \subseteq R_{f_1}$. Suppose that $x_i \notin R_{f_1}$.

Let $N_j = \{x_i, x_{j_1}, x_{j_2}, \ldots, x_{j_t}\}$ be an annuling set of M for f and $x_i \in D(M_1, N_j, f)$. Thus there exists an $t+1$ - tuple $(c_i, c_{j_1}, c_{j_2}, \ldots, c_{j_t}) \in A^{t+1}$ of values for the variables in N_j such that $M_1 \in S_{f_2}$ and $M_1 \notin R_{f_2(x_i=c_i)}$ where

$$f_2 = f(x_{j_1}=c_{j_1}, x_{j_2}=c_{j_2}, \ldots, x_{j_t}=c_{j_t}).$$

Obviously $M \notin S_{f_2}$ and $M_1 \in S_{f_2}$. We will prove that $M_1 \in Max(M, f_2)$. Suppose that there is a subset M_3 of M such that $M_1 \subsetneq M_3$ and $M_3 \in S_{f_2}$. Now, by $f_2 \prec f$ it follows that $M_3 \in S_f$. Hence $M_1 \notin Max(M, f)$. A contradiction. Consequently $M_1 \in Max(M, f_2)$. Theorem 4.2 implies that $M_1 \in S_{f_3}$ and $M_1 \subseteq R_{f_3}$ where

$$f_3 = f_2(x_{m+1}=d_{m+1}, x_{m+2}=d_{m+2}, \ldots, x_p=d_p)$$

Now, by $x_i \notin R_{f_1}$ it follows that

$$f_3 = f_3(x_i=c_i)$$

i.e.

$$f_3 = f(x_i=c_i, x_{j_1}=c_{j_1}, x_{j_2}=c_{j_2}, \ldots, x_{j_t}=c_{j_t}, x_{m+1}=d_{m+1}, x_{m+2}=d_{m+2}, \ldots, x_p=d_p).$$

Consequently $f_3 \prec f_2(x_i=c_i)$ and $M_1 \notin R_{f_2(x_i=c_i)}$ implies $M_1 \notin R_{f_3}$. A contradiction. The proof of the theorem is complete.

Theorem 4.6 is a generalization of Theorem 4.2.

It may be proved following

THEOREM 4.7. If $M \notin S_f$, $M \subseteq R_f$, $M_1 \in Max(M, f)$ and $M \neq \emptyset$ then $Ud(M_1, f) \neq \emptyset$.

MTA SZTAKI Közlemények 35/1986 p.169-180

AN APPLICATION OF CLEVERDON'S TYPE INDICATORS TO EVALUATION OF DATA SECURITY MECHANISM

B. SZAFRAŃSKI

ul. Zawadzkiego 18 m. 38 03-984 Warszawa

Poland

1. INTRODUCTION

Everyone, who comes into contact with the reality of large data base systems agrees, that the data base system environment is complex. Therefore the user, who gives away his data under control of such system, requires the guarantee that his data would be utilized by other users selective according to granted access privileges. In this circumstance the question of data base security system evaluating takes on considerable importance. Basing on the analysis of the publications you can state that this question, except occasional attempts [3,2], is totally ignored. In this paper the proposal of the data owner oriented evaluation method is described. This method relies on the quality indicators, named accessibility C and tihgtness H. These indicators reflect the ability of data base systems to reject unlegal and to serve legal access to data in data base. The accessibility and tightness indicators are an adaptation of Cleverdon's recall R and precision P indicators used to evaluate the information retrieval systems [1]. Using these indicators the paper presents the formal model of evaluation process of data security based on data access control.

2. A MODEL OF DATA ACCESS CONTROL WITH POSSIBILITIES OF EVALUATION

For the resons of examining the quality of data security systems, each name of logic data units was joined with name of the user, who defined the unit and gave it under control of data base system. Such a user was called the owner of the data and would perform the role of an evaluating subject in the evaluation process. It is natural to accept such approach because owner is most interested in the fact, that processing of his data was performed in accordance with their requirements. It is assumed further, that in the evaluation process owners put queries in the form of a pair (data unit name, operation name). The system takes such a query to serve and it makes a dichotomic division of user set into two subsets. The first one is the user name set, which possess the privilege of the access in scope defined by the user-owner in a query and the second one is the user name subset which does not posses such a privilege. The base of defining the quality indicators is the comparison of dividing results of the user name set performed by the system and the results of dividing the same set performed by the owner of the data. Basing on these considerations we can define the data access control model [4].

Definition

The data access control model is a four tuple:

$$AM = \langle U, B, T, \Psi \rangle$$

where: U - the set of user names
B - the set of data unit names
T - the set of operation names
Ψ - accessibility relation ($\Psi \subset U * B * 2^T$).

Accessibility relation determines the user privileges and therefore it joins the user name, data unit name and subset of allowed operations. For example, an element $(u,b,t) \in \Psi$,

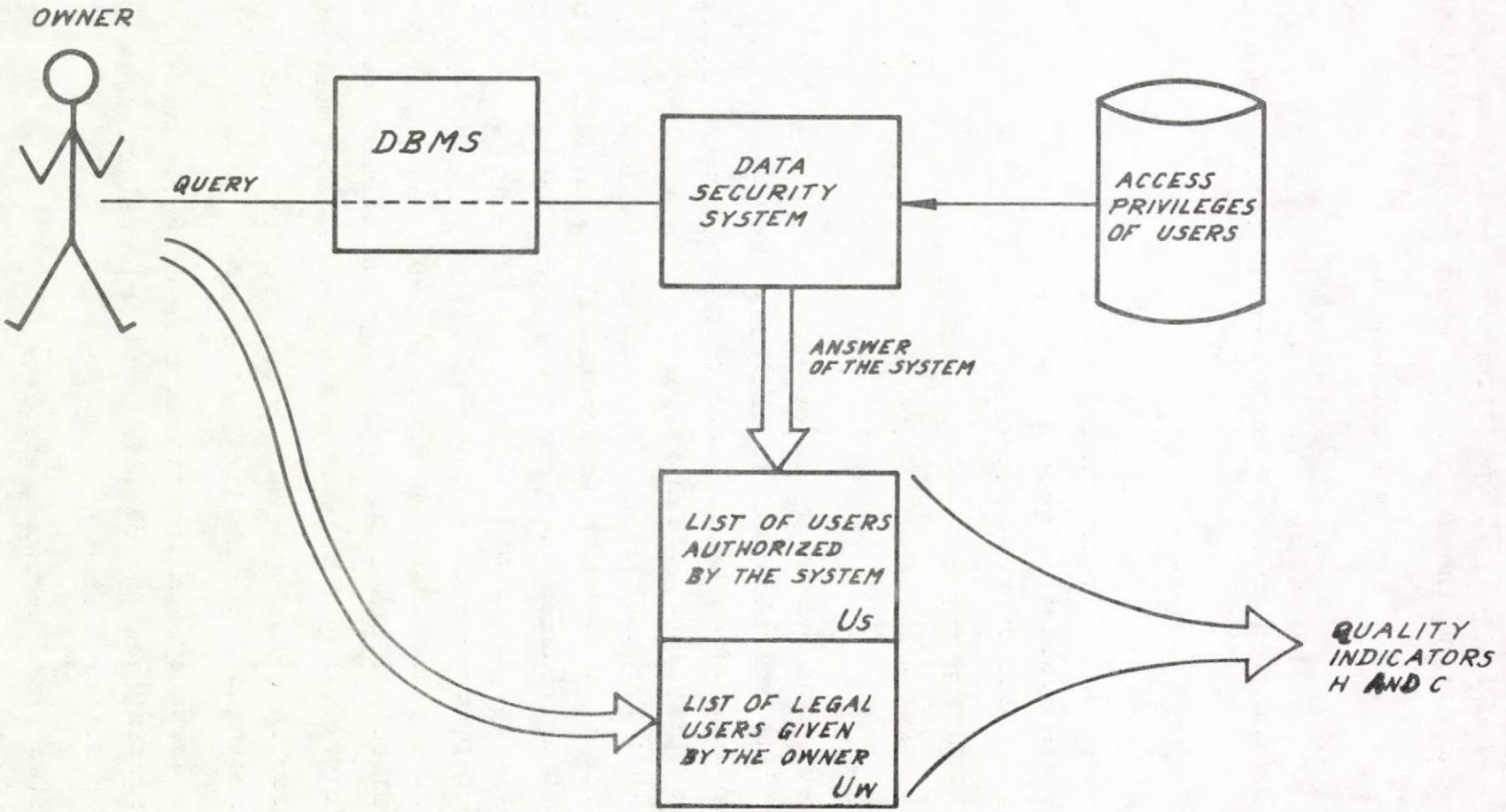


Fig. 1. Schema of data security evaluation

where t = {put,get} means that user u can process data unit b by using only put or get operations.
Now, we can define the evaluation model of data access control:

Definition

The evaluation model of data access control is a triple:

$$EM = \langle AM, V, \rho \rangle$$

where: AM - data access control model,
V - evaluation queries set,
ρ - authorizing function,

$$\rho: V \to 2^{U}, \rho(b,t) = \{u \in U: (u,b,t^*) \in \Psi \; t \quad t^*\} = U_s$$

We can see, that subset U_s is created on the base of accessibility relation and it includes the user names, which have been authorized by the data security system because they had access privilege, which was defined in a query. Let us assume, that for the same query the legal user name subset is known. This subset, which is denoted by U_w is defined by the owner, who introduced query to the system. Finally we have two subsets for the same query: first one, which was determined by data security system and the second one, which was defined by the user-owner. It is important to see, that in general sets reached in such a way can differ from each other. The differences may result from many causes but most impostants are:

- false functioning updating and retrieval procedure concerning accessability relation,
- too great system inertia in case of environment changes (creating new and deleting old objects, changes of authorization scopes),
- software and hardware errors,

- wrong recognition of users and their programs at automatic granting access privileges,
- illegal modifications of the relation by illegal users as a result of insufficient protection of accessibility relation.

Moreover it should be taken into account, that to decrease the design and exploitation costs of data security system the risk of wrong determining the subset U_s by the system (in special cases) could be assumed. Now we can define two quality indicators for each query:

$$C(v) = \begin{cases} \dfrac{|U_w \cap U_s|}{|U_w|} & \text{when} \quad U_w \neq \Phi, \quad U_s \neq \Phi \\ & \text{when} \quad U_s = \Phi, \quad U_w = \Phi \end{cases}$$

$$H(v) = \begin{cases} \dfrac{|U_w \cap U_s|}{|U_s|} & \text{when} \quad U_w \neq \Phi, \quad U_s \neq \Phi \\ 0 & \text{when} \quad U_w = \Phi \vee U_s = \Phi \end{cases}$$

The indicator C is called the accessibility and the indicator H is called the tightness of the data security system. They are determined for the given query $v \in V$, from the data owner's point of view. The values of the indicators C and H are contained in the value set of real numbers in the interval [0,1]. Using this fact profitably the value scale of these indicators has been designated by the order relation in real number set. The number $|Uw \cap Us|$ is the integer of users legal authorized by the access control system, Uw is the number of legal users determined by the data owner and Us is the number of users authorized in the response of data security system (*Fig. 2*).

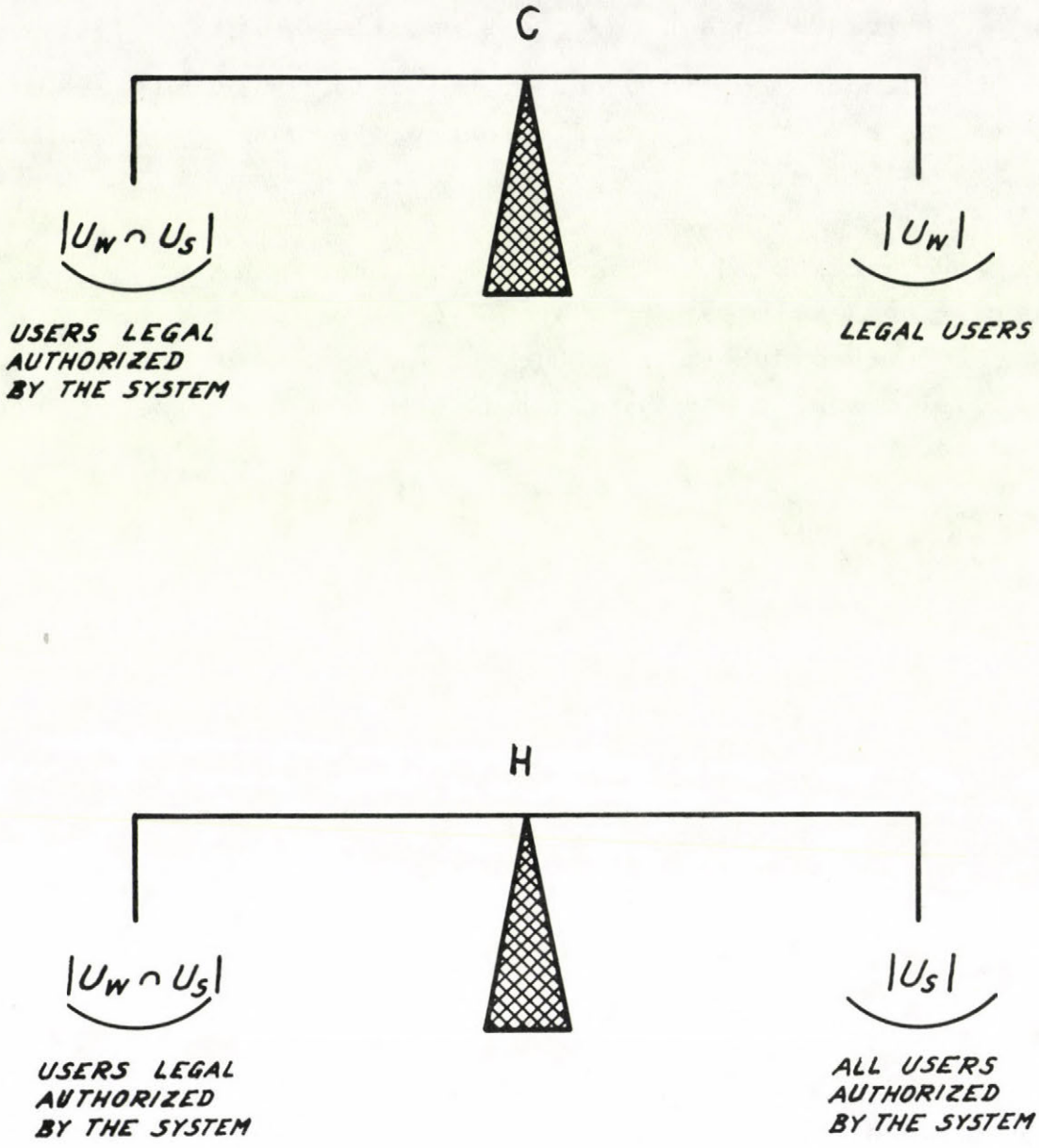


Fig. 2. Illustration of quality indicators

Each of the above mentioned indicators gives another aspect of the division made by the system in comparison with the division made by the user-owner. The accessibility determines the degree of comprising the set of users legal authorized by the system in the legal user set, while the tightness determines the degree of comprising the set of legal authorized users in the set of all users authorized by the system.

The necessity of introducing the indicator C results, among other things, from the requirement that the procedures assuring selective access to data should not cause rejecting of legal demands. For most access control systems this indicator may have an auxiliary sense. Nevertheless its value is great because the efficiency of data base system depends on (among other) quality of data security system. For instance we say, that the probability of terminating the proper and legal task by a small indicator of accessibility is great. However, every groundless terminating of a task diminishes the effectiveness of the system and enlarges its costs. On the other hand the tightness indicator for systems of rigorous requirements on secrecy and property of data has the basic meaning, while it determines the degree (the probability) of proper decisions, which allow to have the access to a defined data.

Let us consider, that if we want to evaluate properly the quality of data access control system, we must know the value of both indicators. The fact, that one of them equals 1 does not forejudge the satisfactory work of the system at all. However, if both of them are equal to 1 ($H = C = 1$) then $Us = Uw$ and then the data access control system works perfectly in the moment of the evaluation (from the user's point of view, who asked the query). We can assume, that reducing the requirements on granting privileges can enlarge the accessibility and intensifying these requirements enlarges the tightness of the data access control system.

3. THE EVALUATION MODEL OF THE OVERALL SYSTEM QUALITY

The previously presented quality indicators make possible the quality evaluation for a particular query put by the data owner. Using these indicators each owner would receive as many value pairs (C,H) as many queries he asked. The sequence of values (C,H) resulting from testing of each query taken from the set of possible queries would be only the input information to draw a conclusion about the quality of the whole data access control system. We have to emphasize, that accessibility and tightness as the features of the system can be indicated to different degree for particular queries of evaluation. Therefore here appears the problem of determining the accessibility and the tightness as overall system quality indicators. The base for drawing conclusions about the overall system quality is the average value of accessibility (tightness) indicator for the given query set. We can use for this purpose typical schemas of average value estimation of the studied property.

Let us assume now, that the query set of evaluation is finite in the moment of evaluation and it has following shape:

$$V = \{v_1, v_2, \ldots, v_i, \ldots, v_I\}$$

Each value C_i is the system response function and the response function of the user-owner to the concrete query v_i. For the studied indicator C (the investigation process of the indicator H follows identically and therefore it will be ommitted) we should evaluate the parameter E[C], which is the average value of the accessibility. Such an indicator is characterized by certain value distribution in the query set. We assume, that this distribution and its parameters are unknown. In connection with it we should gather the random sample to estimate E[C]. As an estimator E[C] we accept the statistics:

$$\bar{C} = \frac{1}{n} \sum_{j=1}^{n} c_j$$

where:

C_j - sample element of $\{C_1, C_2, \ldots, C_j, \ldots, C_n\}$

C_n - is computed from the definition of the indicator C for each j-th drawn query.

$$C_j = \frac{|U_{wj} \cap U_{sj}|}{|U_{wj}|}$$

The condition of sample representativeness $\{C_1, C_2, \ldots, C_j, \ldots, C_n\}$ must be satisfied in result of proper choice of the query sample $\{v_i, v_2, \ldots, v_j, \ldots, v_n\}$ form the set V of possible estimation questions. We can control the choice mentioned above by fitting the size of the sample and using the proper choice schema of the sample. The work [4] presents the adaptation of unrestricted and individual drawing schema of sample elements. However, it is worth to state, that in overall case the choice schema depends on:

- the required evaluation accuracy,
- mutual relation between the owner set and the data set,
- the values of protected data etc.

The stratified drawing schema in which the base of division in strate could be:

- the users-owners, when they differ essentially about the number of data they possess, the knowledge of data base problems or other important a priori known properties,
- data on the score of the importance for the management processes,

• operations, dependently on the sort,

seems to be useful for our purposes.

4. CONCLUSIONS

A. The presented indicators of tightness H and accessibility C can serve directly to make the vectorial (two criteria) evaluation of data security systems. In praxis there is tendency to determine mono-numbering measures (scalar). It seems, that scalar quality measure should be the function of both indicators. The example can be here the measure of the following shape:

$$M = \frac{\text{illegal authorization number + illegal refusal number}}{\text{legal authorization number}} =$$

$$\frac{|U_S| - |U_W \cap U_S| + |U_W| - |U_W \cap U_S|}{|U_W \cap U_S|} = \frac{|U_S| \quad |U_W| - 2 \cdot |U_W \cap U_S|}{|U_W \cap U_S|}$$

After dividing the numerator and denominator by $|U_W \cap U_S|$ and using the definition D and H we receive the shape:

$$M = \frac{1}{C} + \frac{1}{H} - 2$$

B. The indicators discussed above may have various applications. Firstly, they may be applied to compare some properties within one experiment. In such type of studies the query set, user name set and user-owner name set are constant during the experiment but some system parameters are changed (e.g. the access relation updating procedure, the way of protecting and so on).

Secondly, the quality indicators can be used to external comparison of the effects, eg. the comparison of various systems in a certain intervals or the comparison of values C and H in the succeeding intervals or the comparison of values C and H in the succeeding intervals by the same user for the same data to determine the tendencies related to the data security in the defined system. Another typical application of the indicators C and H is their use to estimate the results of experiments simulating the expected properties of access control real systems.

REFERENCES

[1] C. Cleverdon and M. Keen, Factors determining the performance of indexing systems, Test results, vol.2, Aslib-Cranfield Research Project, England, 1966.

[2] L.J. Hoffmann, Modern methods for computer security and privacy, Prentice-Hall, New Yersey, USA, 1977.

[3] R. Summer and C. Wood, Database security and integrity, Addison-Wesley, USA, 1984.

[4] B. Szafrański, The questions of program's data security in database, Diss., MAT, Poland, 1978.

[5] B. Szafrański, A. model of data security systems evaluation in data base, The Third International Seminar on Data Base Management Systems, Zabrow, Poland, 1980.

A Cleverdon tipusu mutatók alkalmazása az adat-biztonsági mechanizmusok értékelésénél

B. Szafranszki

Összefoglaló

A cikk egy modellt mutat be, amely az adat elérés ellenőrzési mechanizmusát értékeli. Az eljárás azokat a mutatókat értékeli, amelyek az adat-tulajdonos szempontjából fontosak, nevezetesen az elérhetőséget és a merevséget /"tightness"/. Ezek a Cleverdon-féle előhivás /"recall"/ és pontosság /"precision"/ mutatók adaptációi. A cikk a rendszer részleges ill. általános minőségének értékelésével is foglalkozik.

Применение индикаторов типа Клевердона к оцениванию механизмов для безопасности данных

Б. Шафрански

Р е з ю м е

Статья показывает модель для оценивания механизма контроля выборки данных. Процедура оценивает те индикаторы, которые важны с точки зрения собственника данных, а именно, возможность доступа к данным и "жесткость" /"tightness"/ данных. Эти являются адаптациями критериев Клевердона "вызов" /"recall"/ и "точность" /"precision"/. Статья занимается тоже оцениванием партикулярного и общего качества системы.

MTA SZTAKI Közlemények 35/1986 p.181-206

SOME PARALLEL ASSOCIATIVE ALGORITHMS FOR IMAGE SEGMENTATION USING THE NR - GRAPHS

GIANG VU THANG

Institute of Computer Science and Cybernetics.
Doithong, Lieugiai, Badinh, Hanoi, Vietnam

I. INTRODUCTION

Up to now, there are various methods for solving the image segmentation problem. One of them is based on the histogram evaluation as presented in our previous work [10]. This algorithm is effective, when the image has bimodal properties. Other methods for solving this problem are based on a large family of the image clustering algorithms. There exist some efficient algorithms e.g. "k mean" algorithms [8] or the algorithms of the type "nuees dynamique" [7]. The result of these algorithms is that some regions of the image with a certain common property are isolated, i.e. the segments of the image are identified.

These algorithms are of iterative nature, they converge and give rather good results after a finite number of passes. However, these algorithms have also some disadvantages, which need to be overcomed. First, by them the spacial information of image pixels haven't been used yet, it is the reason why sometimes the connectedness of the resulting segments of image hasn't been guaranted, in each segment of image there exist some undesired holes. This event makes the next processing procedures difficult. Second, in order to carry out these algorithms, the parameter concerning with the number of image segments must be assigned as a preselected value. The initial assignment of this parameter is based on the experiences of the programmer, therefore, this procedure includes many heuristic and ituitive properties.

In this paper, an other method for solving the image segmen-

tation problem is presented, which depends on the NR-graphic structure of image. First, the definition on the NR-graphic structure of image is given. The number of connected segments of image is found in dependence upon natural image properties. It can be proposed that in each connected segment of the image graph, there uniquely exists one mutual related pair of pixels. This pair can be seen as the kernel of the corresponding segment of image graph. The method developed will use as much as possible the spacial information contained in image pixels for form-ng resulting segments of image. The number of segments equals to the number of the mutual related pairs.

Some parallel algorithms for searching the mutual related pairs and the corresponding connected segments of image are mapped for implementation on the associative processor of the SIMD type. The complexity of each algorithm is investigated. As an illustration, one simple example is examined.

II. BASIC DEFINITIONS AND ASPECTS OF THE THEORY

1. Multigray level image

The multigray level image is represented in term of a square matrix of a size $n \times n$. Where n is the number of image pixels in one column. Each pixel can take one among m possible gray level velues $[0,1,\ldots,m-1]$. An image is assumed to consist of some connected components (or segments) with an assumption that image pixels belonging to the same segment are more "similar" one to another than the pixels lying in different segments of the image.

Let x be an arbitrary pixel of a given image. Let a denote the gray level of pixel x. Further, let a pixel x have coordinates I and J. Then the index IN(x) of pixel x can be defined as

$$IN(x) = (J-1)n + I.$$

The assignment of the index to image pixels is represented in figure 1.

1	n+1	n+2	•	•	••
2	n+2	2n+2	•	•	•
3	n+3	3n+3	•	•	•
•	•	•			

Fig. 1

Definition 1 (Definition of a neighbourhood of a pixel) Giving an arbitrary image pixel xij, Vxij denoted the neighborhood of pixel xij. Vxij consists of eight neighbors of xij (xi-1, j-1, xi-1, j, xi-1, j+1. xi, j-1, xi, j+1, xi+1, j-1, xi+1, j, xi+1, j+1) and of the pixel xij. Neighborhood Vxij of pixel xij is illustrated in the following figure 2

$x_{i-1,j-1}$	$x_{i-1,j}$	$x_{i-1,j+1}$
$x_{i,j-1}$	$x_{i,j}$	$x_{i,j+1}$
$x_{i+1,j-1}$	$x_{i+1,j}$	$x_{i+1,j+1}$

Vxij

Fig. 2

Definition 2. (The distance function between image pixels)
Let x and y are two arbitrary pixels of image with gray level values a and b respectively.

The distance function f is defined as

$$f(x,y) = |a-b|$$

It is easy to notice that, the distance function f defined above satisfies all the possible properties of the metrical function, because

$$1)\quad f(xij,xij) = 0$$

$$2)\quad f(xij,xi'j') = f(xi'j',xij)\ .$$

$$3)\quad f(xij,xi_1j_1)\ f(xij,xi_2j_2)+f(xi_2j_2,xi_1j_1)$$

Definition 3

Let x be an arbitrary image pixel. Then a nearest neighbor of x is a pixel y0 which gives a minimal value of the distance function f in the neighborhood of x. It means that

$$f(x,y0) = \min_{y\in Vx} f(x,y)$$

In some cases, y0 isn't defined uniquely. Then Card (B) ≥ 2

Where $B = \{z \mid z\in Vx,\ f(x,z) = \min_{y\in Vz} f(x,y)\}$.

Then, a nearest neighbor y0 of pixel x is a pixel satisfying the following conditions

$$f(x,y0) = \min_{y\in Vx} f(x,y)$$

and

$$IN(y0) = \max_{z\in B} IN(z)$$

Thus, each pixel has an unique nearest beighbor.

Definition 4

If A is a set of image pixels, the nearest neighbor function NR is a mapping: $A \to A$.

Such that $NR(x) = y$ where y is a nearest neighbor of x. It means that

$$f(x,NR(x)) = \min_{y1\in Vx} f(x,y1)$$

and

$$IN(NR(x)) = \max_{z\in B} IN(z)\ .$$

The nearest neighbor function NR is defined uniquely.

Definition 5. (Definition of the NR-graph of given image.)
The nearest neighbor graph (The NR-graph) of a given image is defined as an ordered pair

$$G = \langle A, V \rangle$$

Where A denotes the set of image pixels and V is a set of oriented arcs of the graph represented in a form

$$V = \{(x,y) \; x,y \in A, \quad y = NR(x)\}.$$

Clearly, The NR-graph G constructed on image A consists of some connected subgraphs Gi, $i = \overline{1,k}$

such that $G = \{Gi \mid i=\overline{1,k}\}$ where $Gi=\langle Ai,Vi\rangle$ $i=\overline{1,k}$,

Each subset Ai of image pixels represents a certain cluster of image. In other words, each subset can be assumed as a segment of image. The subsets Ai, $i = \overline{1,k}$ form segments of image A.

The number of segments is dependent on a concrete content of the initial image and on the gray level distribution. The information of the spacial position and of the similarity of gray levels will be also used for forming the NR-graph.

Theorem 1 Connected subgraph Gi of NR-graph G of image possesse the "tree" pattern i.e. it doesn't exist there any loop of length>2

Proof We have to show that, in a given graph Gi it dcesn't exist any loop with its length>2.

Let us suppose that in Gi there exist a loop $xk1, xi2, \ldots, xin_i$ ($n_i>2$) of length>2 (fig.3)

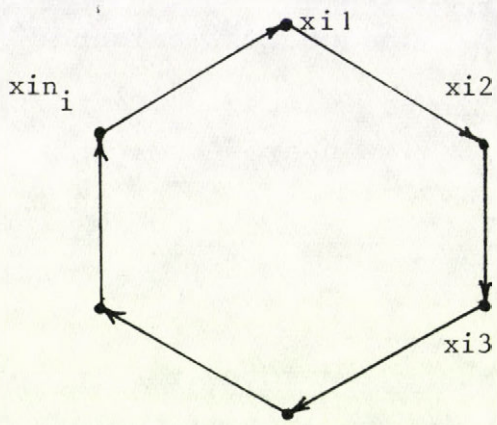


Fig. 3

Let us assume that

$$NR(xi1) = xi2$$

It implies that $NR(xin_i) = xi1$.

Because of $NR(xi1) = xin_i$ then xi1 has two nearest neighbors xi2 and xin_i. It is a contradiction with a unique existence of the nearest neighbor of xi1. Therefore, we have $NR(xin_i) = xi1$

Analogically, if we continue this inference process, we obtain sequence

$$NR(xi1) = xi2$$
$$NR(xi2) = xi3$$
$$\vdots$$
$$NR(xin_i) = xi1$$

From this sequence, we arrive at the following conclusion.

Since $NR(xi2) = xi3$ it implies $f(xi2,xi3) \leq f(xi1,xi2)$

$NR(xi3) = xi4$ it implies $f(xi3,xi4) \leq f(xi2,xi3)$

$\vdots$

$NR(xi1) = xi2$ it implies $f(xi1,xi2) \leq f(xin_i,xi1)$

Therefore, we get

$$f(xi1,xi2) = f(xi2,xi3) = \ldots = f(xin_i,xi1)$$

Without a loss of generality, we suppose that pixel xin_i has the largest index among sequence $xi1,xi2,\ldots,xin_i$, i.e.

$$IN(xin_i) = \max_{k=\overline{1,n_i}} IN(xik)$$

It holds $f(xin_i,xi1) = f(xi1,xi2)$, while $IN(xin_i) > IN(xi2)$

therefore, we get $NR(xi1) = xin_i$. But according to the initial assumption we have $NR(xi1) = xi2$ $(xi2 \neq xin_i)$ and this is a contradiction with the uniquenees of a nearest neighbor. Thus, there doesn't exist any loop of length>2, i.e. this subgraph possesses the "tree" pattern.

Definition 6 (Definition of the mutual related pair)

Let x,y be two pixels of image. We call them as a mutual related pair, if and only if there hold the following relations

$$x = NR(y)$$

$$y = NR(x).$$

Theorem 2 In each connected subgraph $Gi = \langle Ai,Vi \rangle$ $i = \overline{1,k}$ of NR-graph $G = \langle A,V \rangle$, there exists a unique mutual related pair.

Proof

Existence Let us consider a connected subgraph $Gi = \langle Ai,Vi \rangle$ and we show that there exists the mutual related pair.

We suppose that this subgraph consists of n_i nodes corresponding to n_i pixels in image. At first, we mark an arbitrary node of this subgraph (denoted by xi0).

Let xi1 be the nearest neighbor of xi0, i.e. $xi_1 = NR(x0)$, where $xi1 \in Ai$.

Further, xi2 = NR(xi1) and continuing this inference, we obtain the following sequence

$$xi1 = NR(xi0)$$

$$xi2 = NR(xi1)$$

$$xi3 = NR(xi2)$$

$$\bullet$$

$$\bullet \quad \text{where} \quad xi0,xi1,xi2,\ldots \in Ai$$

Since the set Ai is finite (it contains ni nodes), in Gi there must exist a certain node xit ∈ Ai, for which

$$xit + 1 = NR(xit)$$

$$xit + 2 = NR(xit+1)$$

$$\ldots\ldots$$

$$xit = NR(xit+m-1)$$

It means that $xit = NR^m(xit)$, $m \leq ni$ where $NR^m(xit) = NR(NR(...NR(xit))...)$. Thus in the connected subgraph Gi, there must exist one loop of length m ($m \leq ni$).

According to the theorem 1 presented above, there doesn't exist any loop of length > 2, therefore it implies that $m \leq 2$.

Since $xit \neq NR(xit)$ then $m = 2$. Thus, in the connected subgraph Gi of the NR-graph of image, there exists the loop of length 2.

Uniqueness

It remains to demonstrate the uniqueness of the existence of a mutual related pair (it means that the uniqueness of a loop of length 2).

Let there exist two mutual related pairs $(xit,\overline{xit})$ and $(xit,\overline{xit}')$.

Because two mutual related pairs belong to the same connected subgraph Gi, then xit and xit must be connected by the finite number of the arcs (*Fig. 6*)

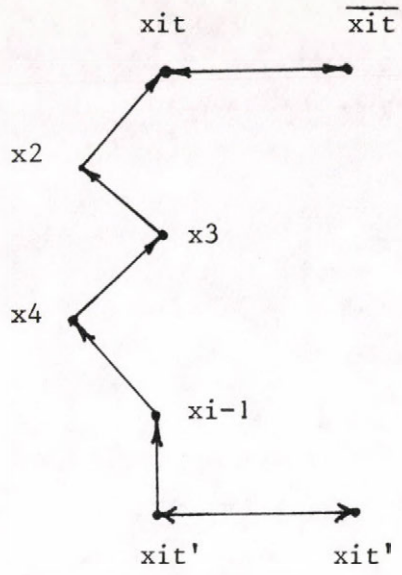


Fig. 6

Let a chain of the nodes connecting xit and xit' is denoted by xit = x1,x2,..., xi = xit'

Because a nearest neighbor of xit is $\overline{xit}$, it implies NR(x2) = xit. Analogically, since a nearest neighbor of x2 is xit it implies NR(x3) = x2.

We get a contradiction that a nearest neighbor of xit' is xi-1 $\neq \overline{xit}'$. Thus pixel xit' has two nearest neighbors which are xi-1 and $\overline{xit}'$. Since one pixel can not have two distinct nearest neighbors. We can conclude that in a connected subgraph there exists one and only one mutual related pair of pixels.

Hence, in each connected subgraph of NR-graph, there exists uniquely one mutual related pair of pixels. We can consider this mutual related pair as a kernel of an image segment assigned to it. The number of the distinct image segments equals to the number of the mutual related pairs.

III. A PARALLEL ASSOCIATIVE ALGORITHM FOR SEARCHING THE CONNECTED COMPONENTS AND THE MUTUAL RELATED PAIRS OF IMAGE

This section brings a method for estimating the number of the mutual related pairs from a sparse matrix structure of the NR relation. This structure is well suited for realization over MDA memory [3].

For a multigray level image, a binary matrix A of the size $n^2 x n^2$ is constructed. Image pixels are ordered in one-dimen-

sional vector. An image pixel xij corresponds to the element xk in one-dimensional vector, where $k = (j-1)n+i$.

Matrix A is calculated according to

$$Aij = \begin{cases} 1 & \text{if } NR(xi) = xj \\ 0 & \text{in other cases} \end{cases}$$

Lemma 1

If in matrix $A^{(h)}$ there exists an element $a^{(h)}_{ij} = 1$ then on the image graph there exists an unique path of length h connecting xi and xj ($A^{(h)} = AxAx...xA$ (h times))

Proof

This lemma is true for $h = 1$.

Let us assume that the lemma is true for $h = k$. We have to prove the lemma for $h = h+1$.

First let us suppose that there exists $a^{(k+1)}_{ij} = 1$. This is equivalent to $\sum_{t=1}^{n} a^{(h)}_{it}\, a_{tj} = 1$. Because each term of the sum is positive, it implies directly that there exists uniquely one term with a certain value t_1 such that

$$a^{(h)}_{it_1} = 1 \quad \text{and} \quad a_{t_1 j} = 1$$

Because of the correctness of the lemma for $h = k$, obviously there exists uniquely one sequence of the nodes of graph xi $xl_1\ xl_2...xl_k = xt_1$ connecting xi and xt_1 of the length k. Therefore, the following sequence xi $xl_1\ xl_2...xl_k$ xj will be a sequence of the length k+1, which connects two pixels xi and sj.

The unique existence of the sequence xi, xl_1, $xl_2...xl_k$, xj is obvious because of the "tree" property of image graph.

Lemma 2

We can choose an relative large integer k1, such that A^{k1} is invariant to multiplication by A^2.

Proof

The proof of this lemma is obvious, because we notice that, during the execution of the multiplication A^k = AxAx...xA (k times) each pixel of image tends to its mutual related pair as k increases.

The desired integer k1 equals to the largest length of subtree among the subtrees of image NR-graph and after k1 multiplications, every pixel has arrived at its mutual related pair. We also notice that the elements of the mutual related pair are transformed to themself by the multiplication with A^2.

Corollary

Let k1 be the least integer described in lemma 2. Then A^{k1+1} is also invariant to A^2 multiplication.

Theorem 3

Let C be $A^{k1}+A^{k1+1}$. Then C is invariant according to the multiplication by A, it means that C = C.A.

Proof

The proof is shown using the results of the lemma and corollary proposed above.

It holds

$$C.A = (A^{k1+1} + A^{k1}).A = A^{k1+2} + A^{k1+1} = A^{k1} + A^{k1+1} = C$$

A. Algorithm (Estimating the value of the invariant matrix C)

In this algorithm, it is assumed that the number of the available processors is n^2 (equal to the number of image pixels). The intermediate fields used in algorithm are

A, A^*, ATG and CS (*Fig. 7*). A is the initial matrix representing the nearest neighbor relation of given image, A^* stores the final result of the matrix multiplications, ATG is a working matrix, the arrays A, A^*, ATG all are the square matrix of a size $n^2 x n^2$. Beside this, CS is an associative field of 2log n-bit width to store the indeces of rows $(1,2,\ldots,n^2)$ from top to bottom.

The main block of algorithm is devoted to estimate A^{n^2} in A^*. The value of A^{n^2} is estimated by a loop of 2log n iterations with a control index k (k = 1,2,..., 2logn). First the

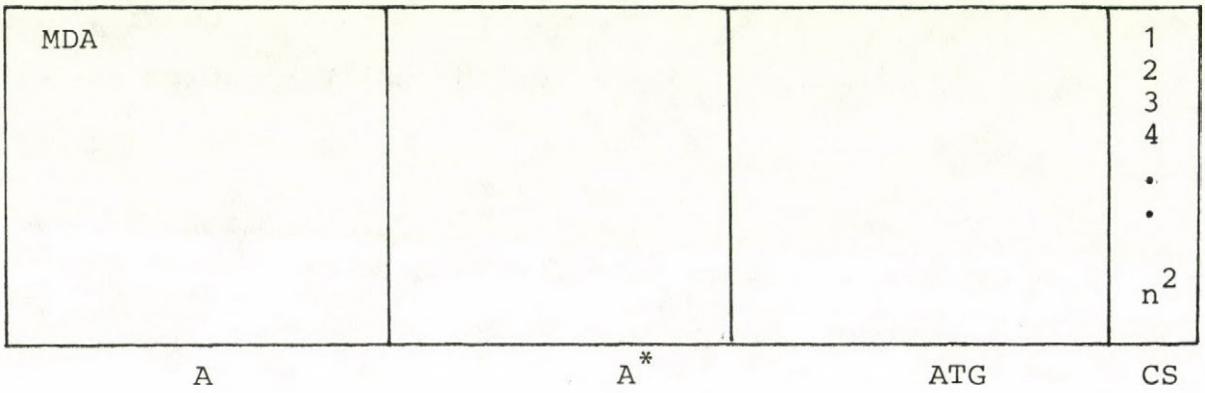


Fig. 7

content of ATG is assigned the value of the initial matrix A, then ATG is multiplicated with itself, the value of A^2 is formed and is stored in A^*. At next iteration of the loop the value of A^4 is formed in A^* etc. and in 2log n steps A^{n^2} is estimated. The next block of the algorithm is devoted to the calculation of $A.A^{n^2}$ and the last one estimates the value of the invariant matrix C located in A^*.

The algorithm can be described in the symbolic form [3] as follows

```
for  j = 1,2,..., n^2
   ATGj ← Aj
Continue
for  k = 1,2,..., 2log n
  for  i = 1,2,..., n^2
    M ← ATGi (bar)
    j ← CS(M)
    Ā*i ← ATGj (bar)
Continue
for  i = 1,2,..., n^2
   ATGi ← A*i
Continue
for  i = 1,2,..., n^2
   M ← ATGi (bar)
   j ← CS(M)
   Ā*i ← Āj
Continue
for  i = 1,2,..., n^2
   A*i ← A*i.OR.ATGi
```

Complexity of algorithm

Counting the complexities of instructions used in algorithm we obtain

$$C1 = 45n^2+2\log n.(n^2(6+c)+5n^2)+n^2(6+c)+6n^2$$

$$\leq 2.(11+c)n^2\log n \qquad = O(n^2\log n)$$

Where c gives the complexity of instruction j ← CS(M)

B. Looking for the number of the connected components of graph

According to theorem 2, the number of the connected components equals to the number of the mutual related pairs. More precisely the mutual element is the one having, at least, one "1" in its corresponding column of the invariant matrix C. Therefire, the algorithm can be written as follows

```
H ← Ø
M ← (1,O,...,O)
for  i = 1,2,...,n^2
   Y ← COMPARE(A*i,I)
   if(COUNT(Y).EQ.O) goto Continue
   H ← (H+1)(M)
Continue
H ← (H/2)
```

Complexity

$$C2 \approx 2\log n+4+n^2(6+9+8\log n) + 4 \log n+6$$
$$\approx 8n^2 \log n = O(n^2 \log n)$$

IV. SOME OTHER PARALLEL ALGORITHM FOR CONSTRUCTING THE NR-GRAPH AND SEARCHING THE MUTUAL RELATED PAIRS FOR THE SIMD TYPE COMPUTER

1. The parallel algorithm constructing the NR-graph of image

a. The main idea of algorithm

Depending on the gray level representation of a given image, the algorithm constructs the matrix representing the NR-graphic

structure of image. The algorithm processes each column of the gray level image from left to right. At each pixel, the distan to the adjac pixels in eight directions ar compared to select a nearest neighbor of the pixel. (The distance is considered as defined in Section 2.)

All pixels of each image column are processed in parallel. As a result the NR-graph of the initial image is obtained.

b. Memory fields used in algorithm

In the algorithm the array AN for storing the initial given image is used.

Output of algorithm is the NR-graph AB of image. The matrix AB contains n columns (AB)i $i = \overline{1,n}$. Each column (AB)i of AB consists of 4 fields of width log n, which are denoted respectively A1i, A2i, B1i and B2i. A1i and A2i hold the row-column coordinates of pixels in i-th column of image. B1i,B2i store the row-column coordinates of the nearest neighbors of the corresponding pixels with coordinates on A1i and A2i respectively.

In this algorithm, we also use some intermediate fields denoted by KCM and TG, where KCM and KCC are log m-bit intermediate fields for representing the distances between pixels of a image column and their nearest neighbors. In addition, TG is the field of the log m-bit width. The representation of the image and of the intermediate fields used are shwon in *Fig. 8.*

Later we shall present the parallel algorithm for constructing the NR-graph of a given image.

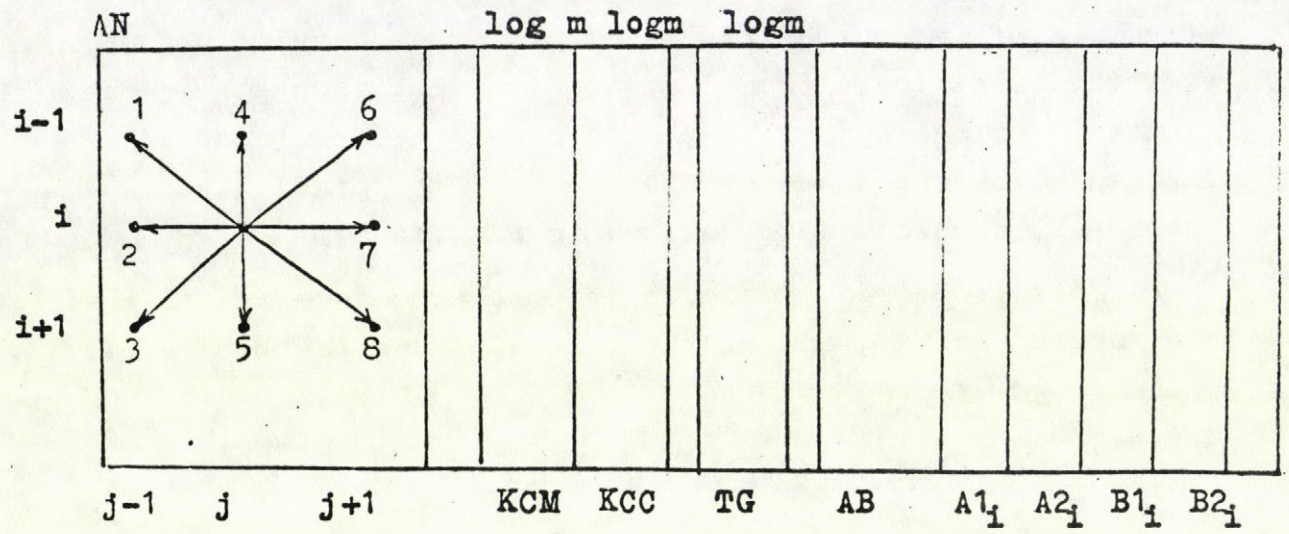


Fig. 8.

c. Algorithm

For j = 1,2,3,...,n

B1j ← $A1_j$ -1

B2j ← $A2_j$ -1

TG ← AN_{j-1}

TG ← $SHIFT^{-1}$(TG)

KCC ← |TG-AN_j|

TG ← $SHIFT^{+1}$(TG)

KCM ← |TG-AN_j|

Y ← COMPARE(KCM.LE.KCC)

M ← Y

KCC ← KCM(M)

$B1_j$ ← $A1_j$(M)

TG ← $SHIFT^{+1}$(TG)

Let us consider pixel xi-1, j-1 as a nearest neighbor of pixel x_{ij}

Consider $x_{i,j-1}$

$KCM \leftarrow TG\text{-}AN_j$

$Y \leftarrow COMPARE(KCM.LE.KCC)$

$M \leftarrow Y$

$KCC \leftarrow KCM(M)$

$B1_j \leftarrow (A1_j+1)(M)$

$TG \leftarrow AN_j$

$TG \leftarrow SHIFT^{-1}(TG)$

$KCM \leftarrow TG\text{-}AN_j$

$Y \leftarrow COMPARE(KCM.LE.KCC)$

$M \leftarrow Y$

$KCC \leftarrow KCM(M)$

$B1_j \leftarrow (A1_j-1)(M)$

$B2_j \leftarrow A2_j(M)$

$TG \leftarrow SHIFT^{+2}(TG)$

$KCM \leftarrow |TG\text{-}AN_j|$

$Y \leftarrow COMPARE(KCM.LE.KCC)$

$M \leftarrow Y$

$KCC \leftarrow KCM(M)$

$B1_j \leftarrow A1_j+1(M)$

$B2_j \leftarrow A2_j(M)$

$TG \leftarrow AN_{j+1}$

$TG \leftarrow SHIFT^{-1}(TG)$

$KCM \leftarrow |TG\text{-}AN_j|$

$Y \leftarrow COMPARE(KCM.LE.KCC)$

$M \leftarrow Y$

Consider $x_{i+1,j-1}$

Consider $x_{i-1,j-1}$

Consider $x_{i+1,j}$

KCC ← KCM(M)

$B1_j$ ← $(A1_j-1)$(M)

$B2_j$ ← $(A2_j+1)$(M)

TG ← $SHIFT^{+1}$(TG)

KCM ← $|TG-AN_j|$

Y ← COMPARE(KCM.LE.KCC)

M ← Y

Consider $x_{i-1,j+1}$

KCC ← KCM(M)

$B1_j$ ← $A1_j$(M)

$B2_j$ ← $A2_j$+1(M)

TG ← $SHIFT^{+1}$(TG)

KCM ← $|TG-AN_j|$

Y ← COMPARE(KCM.LE.KCC)

M ← Y

Consider $x_{i,j+1}$

KCC ← KCM(M)

$B1_j$ ← $A1_j$+1(M)

$B2_j$ ← $A2_j$+1(M)

Consider $x_{i+1,j+1}$

Continue

d. Complexity of algorithm

Counting the complexities of instructions used in algorithm we can estimate the total complexity of algorithm as follows

$$C3 = (361+501\log n+164\log m)n$$
$$= O(n\log n) + O(n\log m)$$
$$= O(n\log k) \quad k = \max(n,m)$$

2. Parallel algorithm looking for the connected segments of image and estimating the number of mutual related pairs on the SIMD associative computer

In this section, an assumption with n^2 processors available is given. Two intermediate fields A,B of the 2long-bit width are to be used. The arrays A_i, B_i $i = \overline{1,n}$ are placed successively in fields A,B from top (see *Fig. 9*)

Later a parallel algorithm for looking for the connected segments of image is to be presented in detail.

Algorithm

In this algorithm, one bit-slice TG is used. It marks the non-leaf vertices of NR graph during the performance of algorithm. First the content of TG is compared with O. That non--leaf which has not yet been processed is marked by content of

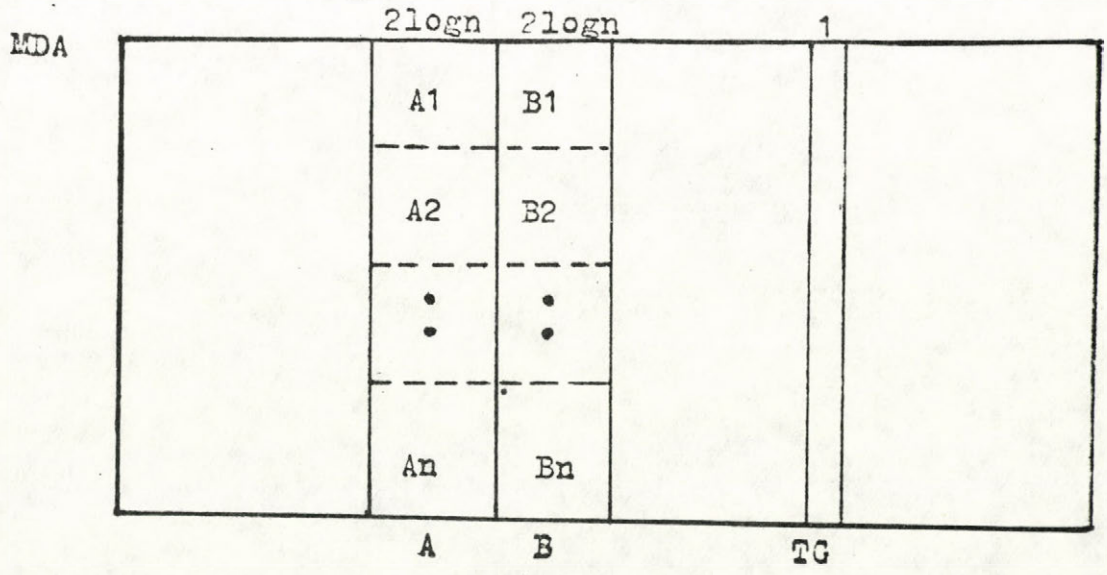


Fig. 9

M register using instruction FIRST RESPONDER(Y). Then index of this node is placed in C register. After that sons of this node are found by a simple searching operation and the fathers of these nodes are replaced by one their grandparent (a father of initial vertex). The places of TG corresponding to non-leaf vertices which have been processed are assigned by 1. In next iterations of algorithm the previous procedure is repeated until the content of the bitslice TG become1. i.e, all possible non-leaf vertices of NR-graph have been processes.

The algorithm can be written as follows

```
        TG ← Ø
N1:     Y ← COMPARE(TG.EQ.Ø)
        M ← FIRST RESPONDER(Y)
        C ← B(M)
        Y ← COMPARE(C.EQ.B)
        X ← Y
        Y ← COMPARE(C.EQ.A)
        M ← Y
        C ← B(M)
        M ← X
        B ← C(M)
        TG ← (TG.OR.I)(M)
        If(TG.EQ.I) STOP
        goto N1
```

Complexity of algorithm

The algorithm is finished, when all the positions of bit-slice TG are set to 1, it means that all non-leaf vertices of the tree-pattern NR graph of image are processed. Therefore, the number of possible iterations of algorithm equals to the number of non-leaf vertices (denoted by n1) in NR graph.

After counting the complexities of instructions used in one iteration, the total complexity of algorithm is given in following form.

$$C4 = 4+n1(6+2+c+3+2\log n+1+3+2\log n+1+c+1+3+6\log n+7)$$
$$= 4+n1(27+2c+10\,1\log n)$$

where c is a constant which gives the complexity of instruction (C ← B(M)). Therefore

$$C4 \approx 10n1\log n = O(n1\log n) \quad (n1 < n^2)$$

Remark in the worst situation when n1 tends to n^2 (the number of image pixels), the holds $C4 = O(n1\log n) \approx O(n^2\log n)$

After the algorithm terminals, all connected segments of image are found. The index in array B belongs to one pixel lying in any mutual related pair. Each pixel in array A belongs to that connected segment which has a kernel in array B in corresponding position. Therefore the number of possible mutual related pairs equals to the number of positions in array A,B where A = B.

Thus, the algorithm counting the number of the mutual related pairs can be expressed in a simple form:

ACKNOWLEDGEMENTS

I am greatly indebted with my thanks to Dr. M. Vajtersic, CSc for his suggestions and kind supervising this work.

The example using the NR-graph
for segmenting the multigray image

0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	2	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
0	0	0	0	4	4	0	0	4	0	0	0	0	0	0	0
0	0	0	0	3	4	0	0	0	0	3	3	3	3	0	0
0	0	0	0	4	4	0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	4	0	0	0	0	0	0	0	4	0	0
0	0	0	0	4	4	3	4	4	4	0	0	0	0	0	0
0	0	0	0	5	4	4	5	4	4	0	0	0	0	0	0
0	0	0	0	4	4	0	0	4	4	0	0	0	0	0	0
0	0	0	0	4	5	0	0	4	4	0	0	0	1	0	0
0	0	0	0	4	3	0	0	5	4	0	0	0	0	0	0
0	0	0	0	5	7	0	0	4	4	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 10 Original image of size 16 x 16 with 8 gray levels (from 0 to 7)

M ← (1,0,...,0)

H ← Ø

Y ← COMPARE (A.EQ.B)

H ← (H + COUNT(Y))(M)

Where H is an associative field of the 2log n-bit width. When the algorithm was finished, the content of field H in first position gives the number of the possible mutual related pairs.

V. THE ILLUSTRATIVE EXAMPLE USING THE NR-GRAPH OF IMAGE FOR SEGMENTING THE MULTIGRAY IMAGE

In this section, we present a simple example for illustrating the performance of the parallel algorithms described above. The original image of a size 16 x 16 with 8 gray levels (from 0 to 7) is given in *Fig. 10*. The resulting segmented image is shown in *Fig. 12* using the results presented.

VI. CONCLUSION.

In this paper, a new approach for the segmentation probelm has been presented. It is based on the definition of the NR-graph of image. Some interesting properties of the image graph are proved, the uniqueness of the mutual related pair in each image segment is pointed out. Some parallel algorithms for looking for the connected segments of image and estimating the number of possible mutual related pairs are described for the associative computer of the SIMD type. First algorithm depending on the multiplications of the sparse matrices obtained the final result in the computational complexity $O(n^2 \lg n)$, while the second one based on the parallel searching operations obtains the same result with the complexity $O(n1 \log n)$ $(n1 \leq n^2)$ where n^2 processors available considered.

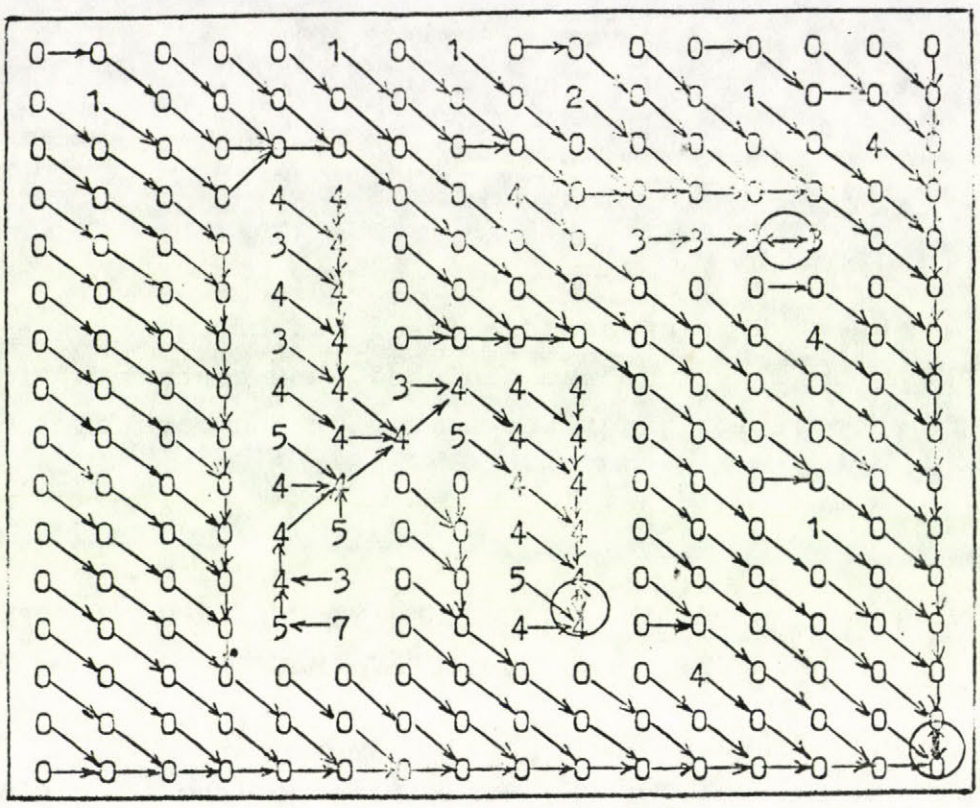


Fig. 11 The NR-graph of a given image.

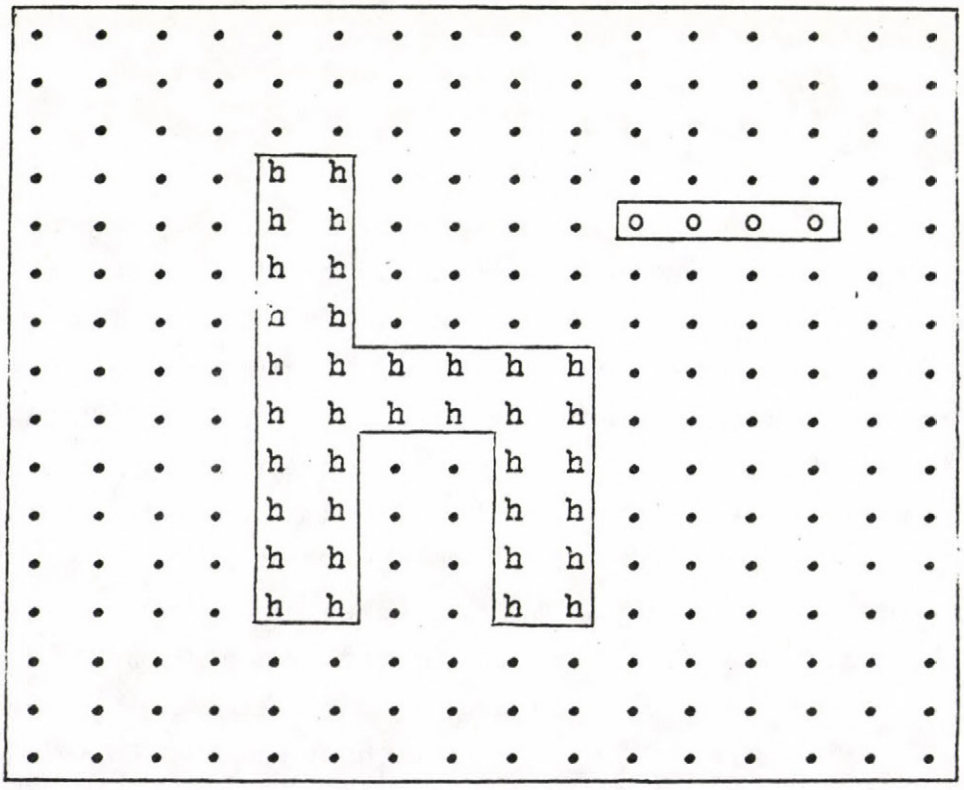


Fig. 12 The segmented image.

REFERENCES

[1] J. Mikloško and V.E. Kotov, Algorithms, Software and Hardware of Parallel Computers. VEDA, Bratislava 1984.

[2] K. Richter, Parallel Computer System SIMD. In proc. AIICSR Conference (I. Plander, ed.) North-Holland 1984, pp 309-313.

[3] J. Mikloško, K. Klette, M. Vajteršic, I. Vrto, Rychle algo-ritmy a ich realizacia n-a špecializovanych počitačoch. VEDA, Bratislava 1985.

[4] W. Blanz et al., Image segmentation by pixel classification, PR 4/81, pp 293-298.

[5] N. Huja et al., Neighbor gray levels as features in pixel classification. PR 4/80, pp 251-266.

[6] D.P. Panda and A. Rosenfeld, Image segmentation by pixel classification in (gray - level, edge value) space. IEEE Trans. Computers C-27, N-9, 1978.

[7] E. Diday et Collaborateurs. Optimisation en Classification Automatique. INRIA, Rocquencourt, 1979.

[8] Tou and Gonzaler.

[9] J. Skákala. ASPL Man, UTK SAV, Bratislava 1981.

[10] M. Vajteršic and Giang Vu Thang. Threshold and Histogram Algorithms for a Parallel Associative Computer. Computer and Artificial Intelligence. Vol. 5(1986) No.2, pp 143-161.

Párhuzamos és asszociativ kép-szegmentációs algoritmusok

Gian Vu Thang

Összefoglaló

A cikkben a szerző egy olyan algoritmust javasol, amely a kép u.n. NR-grafikus strukturájától függ. A módszer lehetőleg minél több, a kép-pontokban levő speciális információt használ fel a szegmensek kereséséhez. Más párhuzamos algoritmusokat is vizsgál a szerző. Az algoritmusok komplexitását is megvizsgálja.

Некоторые параллельные ассоциативные алгоритмы для сегментации изображений, использующие NR-графы

Гиан Ву Тханг

Р е з ю м е

В данной статье представлен новый метод, решающий проблему сегментации изображений. Он основан на концепции NR-графовой структуры данного изображения. Показаны некоторые важные свойства NR-графа изображения. Подробно излагаются два параллельных ассоциативных алгоритма для поиска связанных компонент сегментированного изображения. Первый из них, основанный на перемножении разрешенных матриц, имеет оценку сложности $O/n^2.\log n/$, в то время как второй, использующий операцию параллельного поиска получает тот же результат с оценкой $O/n_1.\log n/$, где $n_1 < n^2$, n - число элементов в данном столбце изображения и доступны n^2 процессоров ЭВМ типа SIMD.

MTA SZTAKI Közlemények 35/1986 p.207-223

SOME RESULTS ABOUT PRIME ATTRIBUTES

VU DUC THI

Computer and Automation Institute
Hungarian Academy of Sciences

1. INTRODUCTION

The prime attributes and minimal keys play essential roles for the normalization of relations. In [5], we have constructed a combinatorial algorithm that from given relation scheme, relation or a closure operation determines a set of all minimal keys.

The antikeys play important roles for the investigation of extremal problems of functional dependencies as well as for the construction of concrete relations representing a set of minimal keys or finding minimal keys. In [4], we constructed a combinatorial algorithm which from given set of minimal keys finds the set of all antikeys. In [2], the equivalence of sets of minimal keys with Sperner-systems has been proved. A set of antikeys is also a Sperner-system.

Lucchesi and Osborn [3] proved that the following prime attribute problem is NP-complete. Given a relation scheme S and an attribute a, decide whether an attributem a belongs tc any minimal key of S. However, in this paper we prove that if $K = \{A_1,\ldots,A_\ell\}$ is the set of minimal keys over Ω and $K^{-1} = \{B_1,\ldots,B_m\}$ is the set of all antikeys of K, then

$\bigcup_{i=1}^{\ell} A_i = \Omega \setminus \bigcap_{i=1}^{m} B_i$, i.e. $\Omega \setminus \bigcap_{i=1}^{m} B_i$ is the set of all prime

attributes. Based on this result we are going to construct an algorithm that decide from a given relation R whether arbitrary attribute is or isn't prime. We prove that worst--case time of our algorithm is polynomial in the number of rows and columns of R. In this paper we give the representation of a minimal key through a set of antikeys.

First we give some necessary definitions, and in Section 2 formulate our results.

Definition 1.1 Let $R = \{h_1,\ldots,h_m\}$ be a relation over the finite set of attributes Ω and $A,B \subseteq \Omega$. We say that B functionally depends on A in R (denote $A \overset{f}{\underset{R}{\rightarrow}} B$) if

$$(\forall h_i,h_j \in R)\,((\forall a \in A)\,(h_i(a) = h_j(a)) \rightarrow (\forall b \in B)\,(h_i(b)=h_j(b))).$$

Let $F_R = \{(A,B): A \overset{f}{\underset{R}{\rightarrow}} B\}$. F_R was called the full family of functional dependencies in R.

Definition 1.2 Let Ω be a finite set, and denote $P(\Omega)$ it's power set. Let $F \subseteq P(\Omega) \times P(\Omega)$. We say that F is a f-family over Ω iff for all $A,B,C,D \subseteq \Omega$

(F1) $(A,A) \in F$;

(F2) $(A,B) \in F, (B,C) \in F \rightarrow (A,C) \in F$;

(F3) $(A,B) \in F,\ A \subseteq C,\ D \subseteq B\ \ (C,D) \in F$;

(F4) $(A,B) \in F,\ (C,D) \in F \rightarrow (A \cup C,\ B \cup D) \in F$.

Clearly, F_R is a f-family over Ω.

It is known [1] that if F is an arbitrary f-family, then there is a relation R over Ω such that $F_R = F$.

Definition 1.3 The mapping $L: P(\Omega) \to P(\Omega)$ is called a closure operation over Ω iff for every $A,B \subseteq \Omega$

(1) $A \subseteq L(A)$,

(2) $A \subseteq B \to L(A) \subseteq L(B)$,

(3) $L(L(A)) = L(A)$.

Remark 1.1 It is easy to see that if F is a f-family, and for all $A \subseteq \Omega$ we set $L_F(A) = \{b \in \Omega: (A,\{b\}) \in F\}$, then L_F is a closure operation over Ω. Conversely, it is proved [1] that if L is a closure operation, there is exactly one f-family over Ω such that $L = L_F$, where $F = \{(A,B): A,B \subseteq \Omega, B \subseteq L(A)\}$.
Thus, between closure operations and f-families over Ω there is one-to-one correspondence.
Clearly, for arbitrary closure operation there is a relation R such that $L = L_{F_R}$.

Definition 1.4 Let R be a relation, L be a closure operation over Ω, and $A \subseteq \Omega$. A is a key of R (a key of L) if $A \overset{f}{\underset{R}{\to}} \Omega (L(A) = \Omega)$. A is a minimal key of R (a minimal key of L) if A is a key of R (a key of L), but $B \overset{f}{\underset{R}{\not\to}} \Omega$ $(L(B) \neq \Omega)$ for any proper subset B of A. Denote $K_R(K_L)$ the set of all minimal keys of R(of L). Clearly, $K_R = K_{L_{F_R}}$ and K_R, K_L are Sperner-systems over Ω.

Definition 1.5 Let K be a Sperner-system over Ω. We define the set of antikeys of K, denoted by K^{-1}, as follows:

$$K^{-1} = \{A \underset{\neq}{\subset} \Omega: (B \in K) \to (B \underset{=}{\not\subset} A) \quad \text{and} \quad (A \underset{\neq}{\subset} C) \to (JB \in K)(B \underset{=}{\subset} C)\}.$$

It is easy to see that K^{-1} is also a Sperner-system over Ω.

Theorem 1.1 ([1,2]). If K is an arbitrary Sperner-system, then there is a closure operation $L(L')$ for which $K = K_L$ $(K = K^{-1}_{L'})$.

In this paper we always assume that if one Sperner-system play the role of the set of minimal keys (antikeys), then this Sperner-system is not empty (doesn't contain Ω).

2. RESULTS

Definition 2.1 Let L be a closure operation over Ω.

Denote $Z(L) = \{A \in P(\Omega) : L(A) = A\}$,

$$T(L) = \{A \underset{\neq}{\subset} \Omega: L(A) = A \text{ and } A \underset{\neq}{\subset} B \to L(B) = \Omega\}$$

The elements of $Z(L)$ are called closed sets. $T(L)$ is called a maximal family of L.

Lemma 2.1 [4] Let L be a closure operation over Ω. Then

$$K_L^{-1} = T(L).$$

Lemma 2.2. Let K be a Sperner-system over Ω and K^{-1} is the set of antikeys of K. Then if $A \in K$, $|A| \geq 2$ (that is, A is a minimal key), then there are $B_i, B_j \in K^{-1}$, and an attribute a $(a \notin B_i)$ such that $A = B_i \cup \{a\}$, and $a \in B_j$.

If $|A| \geq 2$ then by Theorem 1.1 there exists a closure operation L such that $K = K_L$. Let D be the set for which $D \subsetneq A$ and $A \setminus D = a (a \in \Omega)$, i.e. $|D| = |A| - 1$. By Lemma 2.1 and from $A \in K$, we have $L(D) \neq \Omega$, and there is a $B_i (B_i \in K^{-1})$ such that $L(D) \subseteq B$. It is clear that $A \subseteq B \cup \{a\}$. From $|A| \geq 2$, we have $A \subseteq \bigcup_{B_t \in K^{-1}} B_t$. Consequently, there is $B_j \in K^{-1}$ such that $a \in B_j$. The lemma is proved.

Theorem 2.1. Let K be a Sperner-system over Ω and $K^{-1} = \{B_1, \ldots, B_m\}$ is the set of antikeys of $K, A \subseteq \Omega$. Then $A \in K$ holds (that is, A is minimal key) if and only if $A \not\subseteq B_i$ $(\forall i = 1, \ldots, m)$ and for every D $(D \subsetneq A)$ there is a B_i $(B_i \in K^{-1})$ such that $D \subseteq B_i$.

Proof. We suppose that A is the set for which $A \not\subseteq B_i$ $(\forall i = 1, \ldots, m)$ and for every D $(D \subset A)$ there is a B_i $(B_i \in K^{-1})$ such that $D \subsetneq B_i$. Clearly, if $|A| = 1$, then $A \in K$ holds. If $|A| \geq 2$ and there is a $B_i \in K^{-1}$ such that $B_i \subset A$, then A is a key by definition of antikey. If there isn't a $B_i \in K^{-1}$ such that $B_i \subset A$, then $K^{-1} \cup \{A\}$ is a Sperner-system over Ω. Then by Theorem 1.1 there exists a closure operation L so that $K = K_L$. Consequently, if $L(A) \neq \Omega$ then by Lemma 2.1 there is a B_i $(B_i \in K^{-1})$ such that $L(A) \subset B_i$, i.e. $A = B_i$. This contradicts with the fact that $A \not\subseteq B_i$ $(\forall i = 1, \ldots, m)$. That is, $L(A) = \Omega$.

Because for every set $D(D \subsetneqq A)$ there is a B_i $(B_i \in K^{-1})$ such that $D \subseteq B_i$, it is clear that by the definition of minimal key A is a minimal key, i.e. $A \in K$ holds.
On the other hand, we suppose that A is a minimal key, i.e. $A \in K$. It is clear that $A \not\subseteq B_i$ $(\forall i = 1,\ldots,m)$ (by the definition of antikey) and for every D $(D \subsetneqq A)$ if $D \subsetneqq B_i$ $(\forall i = 1,\ldots,m)$ then by the above proof D is a key, which conflicts which the fact that A is a minimal key. Consequently, there exists a B_i $(B_i \in K^{-1})$ so that $D \subseteq B_i$. The theorem is proved.

Definition 2.2 Let $Q = \{A_1,\ldots,A_m\}$ be a family of non-empty sets over Ω, and $|A_i| = \tau_i$ $(\forall i = 1,\ldots,m)$. The following numeration of Q is called primary one:

$$A_1 = \{a_1^1,\ldots,a_{\tau_1}^1\},\ldots,A_i = a_1^i,\ldots,a_{\tau_i}^i\},\ldots,A_m = a_1^m,\ldots,a_{\tau_m}^m.$$

where we choose a_1^1 as follows:

1) Choose a first element a of A_1. If there is an A_i $(1 \leq i \leq m)$ so that $A_i = \{a\}$, then let $a_1^1 = a$. Conversely, we mark all elements a, which occur in A_i-s $(1 \leq i \leq m)$ by the star and

2) We choose a following element a of A_1. If there is an A_i $(1 \leq i \leq m)$ such that a is a unique element of A_i. which was not marked by the star, then let $a_1 = a$. Conversely, we mark all elements a, which occur in A_i-s $(1 \leq i \leq m)$ by the star. The 2-th Step is repeated

until a_1^1 was chosen.

We choose a_1^p $(p = 2,\ldots,m)$ as follows:

In A_p, where P run from 2 to m

3) If there is an a_1^q such that $q < p$ and $a_1^q \in A_p$, then let $a_1^p = a_1^q$. Conversely, we perform a following step:

4) We choose a first element a of A_p, which was not marked by the star. If there is an A_i $(p \leq i \leq m)$ such that a is a unique element of A_i, which was not marked by the star, then $a_1^p = a$. Conversely, we mark all elements a which occur in A_i-s $(p \leq i \leq m)$ by the star. The 4-th Step is repeated until a_1^p was chosen.

5) The 3-th Step is repeated until a_1^m was chosen.

Example 2.1 Let $\Omega = \{1,2,3,4,5\}$ and $A_1 = \{1,2,5\}$, $A_2 = \{1,3,6\}$, $A_3 = \{4,5,6\}$. Then:

$\{1^*,2,5\}$, $\{1^*,3,6\}$, $\{4,5,6\}$;

$\{1^*,2^*,5\}$, $\{1^*,3,6\}$ $\{4,5,6\}$; because 5 is a unique element of A_1 which was not marked by the star, we set $a_1^1 = 5$ and

$\{5,1^*,2^*\}$, $\{1^*,3^*,6\}$, $\{4,5,6\}$; because 6 is a unique element of A_2 which was not marked by the star, we have $a_1^2 = 6$ and

$\{5,1^*,2^*\}$, $\{6,1^*,3^*\}$, $\{4^*,5,6\}$; because $5 = a_1^1$ and $5 \in A_3$, we have $a_1^3 = 5$. That is, $A_1 = \{5,1,2\}$, $A_2 = \{6,1,3\}$ and $A_3 = \{5,4,6\}$ is a primary numeration.

Remark 2.1 It is wasy to see that for a given family Q of non-empty sets over Ω it is possible that there are many primary numerations of Q. For the constructing of primary numeration we only construct a_1^i-s $(i = 1,\ldots,m)$. Clearly, the construction of primary numeration depends on the order of elements of A_i $(i = 1,\ldots,m)$, i.e. if the order of elements of A_i $(i = 1,\ldots,m)$ is changed them we can obtain a new primary numeration.

Remark 2.2 By Theorem 2.1 it is easy to see that if $a \in \Omega$, then $\{a\} \in K$ if and only if $a \notin \bigcup_{i=1}^{m} B_i$, where $K^{-1} = \{B_1,\ldots,B_m\}$.

Theorem 2.2 Let K be a Sperner-system over Ω, and $K = \{B_1,\ldots,B_m\}$ is the set of antikeys of K. If a is an attribute such that there is a sequence $(\pi_o,\pi_1,\ldots,\pi_k)$ $(k \geq 1)$ which is an ordered subset of the indices $1,\ldots,m$, and $a \in B_{\pi_i}$ $(\forall i = 1,\ldots,k)$, $a \notin B_j (\forall j : j \notin \{\pi_1,\ldots,\pi_k\})$, then $A = \bigcup_{i=1}^{k} \{a_1^{\pi_i}\} \cup \{a\}$ is an element of K, where $\{B_{\pi_o} \cap \bar{B}_{\pi_i} = \{a_1^{\pi_i},\ldots,a_{\tau_{\pi_i}}^{\pi_i}\}\}$ $(i = 1,\ldots,k)$ is primary numeration. Clearly, $|A| \geq 2$.

Proof. It is easy to see that by $A \cap (B_{\pi_o} \cap \bar{B}_{\pi_i}) \neq \emptyset$ $(\forall i = 1,\ldots,i)$ we have $A \not\subseteq B_{\pi_i}$ $(\forall i = 1,\ldots,k)$. From $a \notin B_j$ $(\forall j : j \notin \{\pi_1,\ldots,\pi_k\})$ we have $A \not\subseteq B_j$ $(\forall B_j \in K^{-1} \setminus \{B_{\pi_1},\ldots,B_{\pi_k}\})$. Consequently, $A \not\subseteq B_i$ $(\forall i = 1,\ldots,m)$. Clearly $\bigcup_{i=1}^{k} \{a_1^{\pi_i}\} \subseteq B_{\pi_o}$ holds. By the definition of primary numeration for every π_i $(1 \leq i \leq k)$ there is a π_j such that $A \cap (B_{\pi_o} \cap \bar{B}_{\pi_j}) = \{a_1^{\pi_i}\}$. Consequently, if D is a proper subset of A such that $a \in A \setminus D$, then $D \subseteq B_{\pi_o}$. Clearly, if D is a proper subset of A so that $a \in D$ and $a_1^{\pi_i} \in A \setminus D$ $(1 \leq i \leq k)$, then there is a π_j so that $A \cap (B_{\pi_o} \cap \bar{B}_{\pi_j}) = \{a_1^{\pi_i}\}$. Consequently, from $a \in D$ and $\bigcup_{p=1}^{k} \{a_1^{\pi_p}\} \subseteq B_{\pi_o}$ we have $D \subseteq B_{\pi_j}$. By Theorem 2.1 $A \in K$ holds. The theorem is proved.

Theorem 2.3. Let K be a Sperner-system over Ω and $K^{-1} = \{B_1,\ldots,B_m\}$ is the set of antikeys of K. If $A \in K$ and $|A| \geq 2$, then there exists an attribute a and a primary numeration $\{B_{\pi_o} \cap \bar{B}_{\pi_i} = a_1^{\pi_i},\ldots,a_{\tau_{\pi_i}}^{\pi_i}\}$, $1 \leq i \leq k$, $k \geq 1$ such that $(\pi_o,\pi_1,\ldots,\pi_k)$ is an ordered subset of the indices $1,\ldots,m$, $a \in B_{\pi_i}$ $(\forall i = 1,\ldots,k)$, $a \notin B_j$ $(\forall j : j \notin \{\pi_1,\ldots,\pi_k\})$, and $A = \bigcup_{i=1}^{k} \{a_1^{\pi_i}\} \cup \{a\}$ holds.

Proof If $|A| \geq 2$ and $A \in K$, then by Lemma 2.2 there is $a \in \Omega$ and $aB_i (B_i \in K^{-1})$ such that $A \subseteq B_i \cup \{a\}$, where $a \notin B_i$.

Denote $\{B_{\pi_1},\ldots,B_{\pi_k}\} = \{B_j \in K^{-1} : a \in B_j\}$.

It can be seen that $\{B_{\pi_1},\ldots,B_{\pi_k}\} \neq \emptyset$ holds, i.e. $k \geq 1$. We set $\pi_o = i$. It is clear that $(\pi_o,\pi_1,\ldots,\pi_k)$ is an ordered subset of the indices $1,\ldots,m$.

By K^{-1} is a Sperner-system over Ω we have $B_{\pi_o} \cap \bar{B}_{\pi_i} \neq \emptyset$. It is clear that $a \in B_j$ $(\forall j : j \notin \{\pi_1,\ldots,\pi_k\})$. Because $A \in K$ and by the definition of antikey $A \cap B_{\pi_o} \cap \bar{B}_{\pi_i} \neq \emptyset$ $(\forall i = 1,\ldots,m)$. It is easy to see that from $A \in K$ and by Theorem 2.1 for every b $(b \in A\backslash\{a\})$ there is a π_i such that $A \cap B_{\pi_o} \cap \bar{B}_{\pi_i} = \{b\}$. Consequently, we can suppose that for every i such that $((B_{\pi_o} \cap \bar{B}_{\pi_i}) \cap A) \neq \emptyset$. $B_{\pi_o} \cap \bar{B}_{\pi_i}$ has a following form: $B_{\pi_o} \cap \bar{B}_{\pi_i} = \{a_{i_1},\ldots,a_{i_\tau}\}$ and $A \cap B_{\pi_o} \cap \bar{B}_{\pi_i} = \{a_{i_j}, a_{i_{j+1}},\ldots,a_{i_\tau}\}$, where $j \leq \tau$. Thus, the elements of A stand in the last places of $B_{\pi_o} \cap \bar{B}_{\pi_i}$. After that we construct a primary numeration of $\{B_{\pi_o} \cap \bar{B}_{\pi_i} : 1 \leq i \leq k\}$.

Denote $C = \bigcup_{i=1}^{k} \{a_1^{\pi_i}\} \cup \{a\}$. It is obvious that $a \in A$.

Clearly, from the construction of primary numeration, we have $a_1^{\pi_i} \in A$ $(\forall i = 1,\ldots,k)$. Consequently, $C \subseteq A$.

On the other hand by Theorem 2.2 C is a minimal key, i.e. $C \in K$. Hence $C = A$ holds. The theorem is proved.

Denote $a_1^{\pi_o}$ the element a. From Remark 2.2, Theorem 2.2 and theorem 2.3 we have a following representation of a minimal minimal key A, i.e. $A \in K : A = \bigcup_{i=o}^{k} \{a_1^{\pi_i}\}$, where $k \geq 0$.

• *Theorem 2.4* Let $K = \{A_1,\ldots,A_\ell\}$ be a Sperner-system over Ω and $K^{-1} = \{B_1,\ldots,B_m\}$ is the set of antikeys of K. Then

$$\bigcup_{i=1}^{\ell} A_i = \Omega \setminus \bigcap_{i=1}^{m} B_i.$$

Proof. We suppose that $c \in \bigcup_{i=1}^{\ell} A_i$ $(A_i \in K)$.

Hence $\exists A_p$ $(1 \leq p \leq \ell$ and $A_p \in K) : c \in A_p$. By Remark 2.2 it can be seen that if $|A_p| = 1$, then $c \notin \bigcup_{i=1}^{m} B_i$, i.e. $c \in \bigcap_{i=1}^{m} B_i$ $(B_i \in K^{-1})$. If $|A_p| \geq 2$, then by Theorem 2.3 there is on attribute a and a primary numeration

$\{B_{\pi_o} \cap \bar{B}_{\pi_i} = \{a_1^{\pi_i},\ldots,a_{\tau_{\pi_i}}^{\pi_i}\}, 1 \leq i \leq k, k \geq 1\}$ such that $(\pi_o,\pi_1,\ldots,\pi_k)$ is an ordered subset of the indices $1,\ldots,m$, $a \in B_{\pi_i}$ $(\forall i = 1,\ldots,k)$, $a \notin B_j$ $(\forall j : j \notin \{\pi_1,\ldots,\pi_k\})$ and $A_p = \bigcup_{i=1}^{k} \{a_1^{\pi_i}\} \cup \{a\}$. Then if $c = a$ holds, then $c \notin B_{\pi_o}$, i.e. $c \notin \bigcap_{i=1}^{m} B_i$ hold. If $c = a_1^{\pi_i}$ holds, then by $a_1^{\pi_i} \in B_{\pi_o} \cap \bar{B}_{\pi_i}$ we have $c \notin B_{\pi_i}$, i.e. $c \notin \bigcap_{i=1}^{m} B_i$ $(B_i \in K^{-1})$. Hence $c \notin \bigcap_{i=1}^{m} B_i$ holds. Thus, $\bigcup_{i=1}^{\ell} A_i \subseteq \Omega \setminus \bigcap_{i=1}^{m} B_i$ holds.

It is obvious that $\bigcup_{i=1}^{\ell} A_i = \Omega \setminus \bigcap_{i=1}^{m} B_i$. The theorem is proved.

Definition 2.3 Let Ω be a non-empty finite set, and $R = \{h_1,\ldots,h_m\}$ be a relation over Ω. Let E_R is the equality set of R, i.e. $E_R = \{E_{ij} : 1 \leq i \leq j \leq m\}$, where $E_{ij} = \{a \in \Omega : h_i(a) = h_j(a)\}$.

Let $M_R = \{A \subsetneq \Omega : \exists E_{ij} \in E_R : E_{ij} = A$ and

$\bar{\exists} E_t \not\equiv R : A \subsetneq E_t : 1 \leq i \leq j \leq m, \quad 1 \leq \ell < t \leq m\}$

M_R is called the maximal equality system of R.

Definition 2.4 [1] Let F be a f-family over Ω, and $(A,B) \in F$. We say that (A,B) is a maximal right side dependency of F iff $\forall B'(B \subsetneq B') : (A,B') \in F \rightarrow B' = B$. Denote by $M(F)$ the set of all maximal right side dependencies of F. We say that B is a maximal side of F iff there is an A so that $(A,B) \in M(F)$. Denote $I(F)$ the set of all maximal sides of F.

Definition 2.5([1,2]) Let K be a non-empty Sperner--system and R be a relation over Ω. We say that R represents K if $K_R = K$, where K_R is the set of all minimal keys of R.

Theorem 2.5 Let K be a non-empty Sperner-system, and R be a relation over Ω. Then R represents K iff $K^{-1} = M_R$, where K^{-1} is the set of antikeys of K and M_R is the maximal

equality system of R.

Proof. Because K is a non-empty Sperner-system, K^{-1} exists. On the other hand, K and K^{-1} are uniquely determined by each other, we have $K_R = K$ holds iff $K_R^{-1} = K^{-1}$ holds. Consequently, we must proved that $K_R^{-1} = M_R$ holds.
It is obvious that F_R is a f-family. Now, we suppose that A is an antikey of K_R, i.e. $A \in K_R^{-1}$. It can be seen that $A \neq \Omega$. If there exists a B such that $A \overset{R}{\subseteq} B$ and $A \overset{f}{\underset{R}{\rightarrow}} B$, then by the definition of antikey we have $B \overset{f}{\underset{R}{\rightarrow}} \Omega$. Hence $A \overset{f}{\underset{R}{\rightarrow}} \Omega$ holds. This contradicts to $\forall c \in K_R : c \not\subseteq A$.
So $A \in I(F_R)$ holds. If there is a B' so that $B' \neq \Omega$, $B' \in I(F_R)$ and $A = B'$, then B' is a key of R. This contradicts to $B' \neq \Omega$. Consequently, $A \in I(F_R)\backslash\{\Omega\}$ and $\bar{\exists} B' \ (B' \in I(F_R)\backslash\Omega) : A \subsetneq B'$. On the other hand, according to the definition of relation $\Omega \notin M_R$. It is easy to see that $E_{ij} \in I(F_R)$. Thus, $M_R \subseteq I(F_R)$ holds. If D is a set so that $\forall c \in M_R : D \not\subseteq C$, then by the definition of functional dependency D is a key of R. Consequently, M_R is the set of maximal distinct elements of $I(F_R)$. So we have $A \in M_R$.
Consequently, we assume that $A \in M_R$. According to the definition of relation and M_R we obtain $A \overset{f}{\underset{R}{\not\rightarrow}} \Omega$, i.e. $\forall B \in K_R : B \not\subseteq A$. On the other hand because A is a maximal equality set, for all D $(A \subsetneq D)$ $D \overset{f}{\underset{R}{\rightarrow}} \Omega$ holds. Consequently, by the definition of antikey $A \in K_R^{-1}$ holds.
The theorem is proved.

Definition 2.6 Let R be a relation over Ω and K_R be a set of minimal keys of R. We say that an attribute a is prime of R if there is a $A \in K_R$ such that $a \in A$.

Based on Theorem 2.4 and Theorem 2.5 we construct a following algorithm.

Algorithm 2.1 Let $R = \{h_1,\ldots,h_m\}$ be a relation over $\Omega = \{a_1,\ldots,a_m\}$.

Step 1 From given a relation R we construct the set $E_R = \{E_{ij} : 1 \leq i < j \leq m\}$, where $E_{ij} = \{a \in \Omega : h_i(a) = h_j(a)\}$.

Step 2 From E_R we construct the set $T = \{A \subsetneq \Omega: \exists E_{ij} \in E_R : E_{ij} = A\}$.

Step 3 From T we construct the set $M_R = \{A \in T: \bar{\exists} B \in T: A \subsetneq B\} = \{A_1,\ldots,A_R\}$.

Step 4 We construct the set $V = \Omega^R \setminus \bigcap_{i=1}^{k} A_k$.

Remark 2.3 In algorithms we assume that the elementary step being counted is the comparison of two attributes. Thus, if we assume that subsets of Ω are represented as sorted lists of attributes, then a Boolean operation on two subsets of Ω requires at most $|\Omega|$ elementary steps.

Proposition 2.1 Let $R = \{h_1,\ldots,h_m\}$ be a relation over $\Omega = \{a_1,\ldots,a_m\}$.

Let K_R is the set of minimal keys of R and $K_R = \{B_1,\ldots,B_\ell\}$.

$K_R^{-1} = \{A_1,\ldots,A_k\}$ is the set of antikeys of R. Then

Algorithm 2.1 determines the set $V = \bigcup_{i=1}^{\ell} B_i$, and worst-case time of this algorithm is polynomial in the number of rows and columns of R.

Proof. Clearly, $|M_R| \leq |T| \leq |E_R| = \binom{m}{2}$ holds. Consequently, the time complexity of Algorithm 2.1 is polinomial in the number of rows and columns of R. By Theorem 2.5 we have $M_R = K_R^{-1}$. By Theorem 2.4 $V = \Omega \setminus \bigcap_{i=1}^{k} A_i = \bigcup_{i=1}^{\ell} B_i$ holds. The proposition is proved.

It is easy to see that based on Algorithm 2.1 from a given relation R we decide whether arbitrary attribute a is prime (that is, $a \in \bigcup_{i=1}^{\ell} B_i$) or not.

The following corollary is obvious.

Corollary 2.1 There exists an algorithm that from a given relation R, decide whether arbitrary attribute a is prime one of R or not, and worst-case time of this algorithm is polinomial in the number of rows and columns of R.

ACKNOWLEDGEMENT

The author would like to take this opportunity to express deep gratitude to Professor Dr. János Demetrovics for his help, valuable comments and suggestions.

REFERENCES

[1] Armstrong, W.W., Dependency Structures of Data Base Relationships. Information Processing 74, North-Holland Pub. Co. (1975) 580-583.

[2] Demetrovics, J., On the equivalence of candidate keys with Sperner-systems. Acta Cybernetica 4(1979) 247-252.

[3] Lucchesi, C.L. and Osborn, S.L., Candidate keys for Relations. Journal of Computer and System Sciences 17, (1978) 270-279.

[4] Thi, V.D., Minimal keys and anti keys. Acta Cybernetica Tom7. Fasc. 2. (1986) 261-271.

[5] Vu Duc Thi, Algorithms for finding minimal keys and anti keys of relational data base. MTA SZTAKI Közlemények, 33 (1985) 113-143.

Eredmények a prim-attributumokra vonatkozóan

Vu Duc Thi

Összefoglaló

A szerző bebizonyitja, hogy ha egy relációs adatbázisban $K = \{A_1,\ldots,A_\ell\}$ a minimális kulcsok halmaza Ω felett, és $K^{-1} = \{B_1,\ldots,B_m\}$ az anti-kulcsok halmaza, akkor $\bigcup_{i=1}^{\ell} A_i = \Omega \setminus \bigcap_{i=1}^{m} B_i$, azaz $/\Omega \setminus \bigcap_{i=1}^{m} B_i/$ a prim-attributumok /"prime-attributes"/ halmaza. Felhasználva ezt az eredményt, algoritmust ad annak eldöntésére, vajon egy adott R relációra vonatkozó attributum prim-e vagy sem. Algoritmusa az R reláció sorainak és oszlopainak számára nézve polinomiális.

Результаты касющиеся прим-атрибутов

Ву Диц Тхи

Р е з ю м е

Пусть в реляционной базе данных $K = \{A_1,A_2,\ldots,A_\ell\}$ есть множество минимальных ключей /над Ω/, а $K^{-1} = \{B_1,\ldots,B_m\}$ множества анти-ключей. В статье доказано, что $\bigcup_{i=1}^{\ell} A_i = \Omega \setminus \bigcap_{i=1}^{m} B_i$, другими словами $/\Omega \setminus \bigcap B_i/$ есть множество прим-атрибутов. Используя этот результат, автор дает алгоритм для решения вопроса, что данный атрибут есть прим или нет. Число шагов алгоритма есть полиномиально в числе столбцов и рядов в реляции R.

Készült az Országos Széchényi Könyvtár
Sokszorosító üzemében, Budapest
Felelős vezető: Rosta Lajosné
Példányszám: 240
Terjedelem: 28,25 A/5 ív
Munkaszám: 87.049